REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-01-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden to Department of Defense, Washington Headquarters Services Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
12–2004	Final Report	June 2001 through December 2004

4. TITLE AND SUBTITLE

AUTOMATED DESIGN TOOLS FOR INTEGRATED MIXED SIGNAL MICROSYSTEMS (NeoCAD) FINAL REPORT

5a. CONTRACT NUMBER
N66001-01C-8042

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHORS

P. Petre, J. Visher, R. Shringarpure, F. Valley, M. Swaminathan
HRL Laboratories, LLC

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

HRL Laboratories, LLC
3011 Malibu Canyon Road
Malibu, CA 90265

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

SSC San Diego
San Diego, CA 92152-5001

10. SPONSOR/MONITOR'S ACRONYM(S)
SSC San Diego

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Automated design tools and integrated design flow methodologies were developed that demonstrated more than an order-of-magnitude reduction in cycle time and cost for mixed signal (digital/analog/RF) and mixed electronic-photonic microsystems, such as high-clock-rate and wide-dynamic-range Analog-to-Digital (A/D) converters, These tools and methodologies combined several innovative synthesis, simulation, and optimization techniques, including a fast time-domain circuit simulation method, Model Order Reduction (MOR) tools, system-level, mixed-signal circuit synthesis and optimization tools, and parsitic extraction tools. A unique combination of these innovative techniques enables efficient design of systems far beyond today's state-of-the-art. In the future, these tools will allow designers of high-performance, mixed-signl circuits, such as high-order tunable, delta-sigma modulators and the photonic A/D converters operating at 10s of GHz clock rates, to optimize their designs with orders of magnitude more iterations, and hence, far better performance.

15. SUBJECT TERMS

Mission Area: Command and Control

mixed signal	circuit simulation	parasitic extraction	time-domain simulation
IC design flow	model order reduction	hardware description language	

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			C. Hanson
U	U	U	UU	138	19B. TELEPHONE NUMBER *(Include area code)* (619) 553-5242/2164

February 2005

Automated Design Tools for Integrated Mixed-Signal Microsystems (NeoCAD) Final Report

L. P. Petre
J. Visher
R. Shringarpure
F. Valley
M. Swaminathan
HRL Laboratories, LLC

20061130032

Approved for public release;
distribution is unlimited.

SSC San Diego

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

Automated design tools and integrated design flow methodologies were developed that demonstrated more than an order-of-magnitude reduction in cycle time and cost for mixed signal (digital/analog/RF) and mixed electronic-photonic microsystems, such as high-clock-rate and wide-dynamic-range Analog-to-Digital (A/D) converters. These tools and methodologies combined several innovative synthesis, simulation and optimization techniques. These techniques include

- A fast time domain circuit simulation method
- Model Order Reduction (MOR) tools
- System-level mixed-signal circuit synthesis and optimization tools
- Parasitic extraction tools

A unique combination of these innovative techniques enabled efficient design of systems far beyond today's state-of-the-art. These tools in the future will allow designers of high performance mixed signal circuits, such as high-order tunable delta-sigma modulators and the photonic A/D converters operating at 10's of GHz clock rates, to optimize their designs with orders of magnitude more iterations and hence far better performance.

1. 1 Improved Mixed-Signal IC Design Flow

Conventional mixed-signal circuit design proceeds at three levels as shown in Fig.1. The top level (i.e., Level 1) is the architectural level where the behavioral simulation, circuit synthesis and system level optimization are performed. The middle level (i.e., Level 2) contains the detailed equivalent circuits for all of the building blocks found in the architectural level. The bottom level (i.e., Level 3) contains the layouts of the circuit blocks. At present, the circuit designer works back and forth between Level 2 and Level 1 to produce a reduced order model of the circuits. Usually, this effort requires considerable intuition on the part of the circuit designer. Unfortunately, a wide range of parasitic processes are also produced as part of laying out the circuit in Level 3. These processes must be included in the Level 2 circuit models and incorporated back into the Level 1 synthesis and optimization.

With the new HRL CAD tool framework the ad hoc methodology described above was replaced with a consistent and seamless design flow. First, the link between Level 3 and Level 2 and that between Level 2 and Level 1 was performed automatically with reduced order models. Secondly, an additional reduced order model linked Level 3 directly to Level 1. Thirdly, the parasitics in Level 3 were computed with a fast method. These steps allowed synthesis to occur directly at Level 1, including all the parasitics.

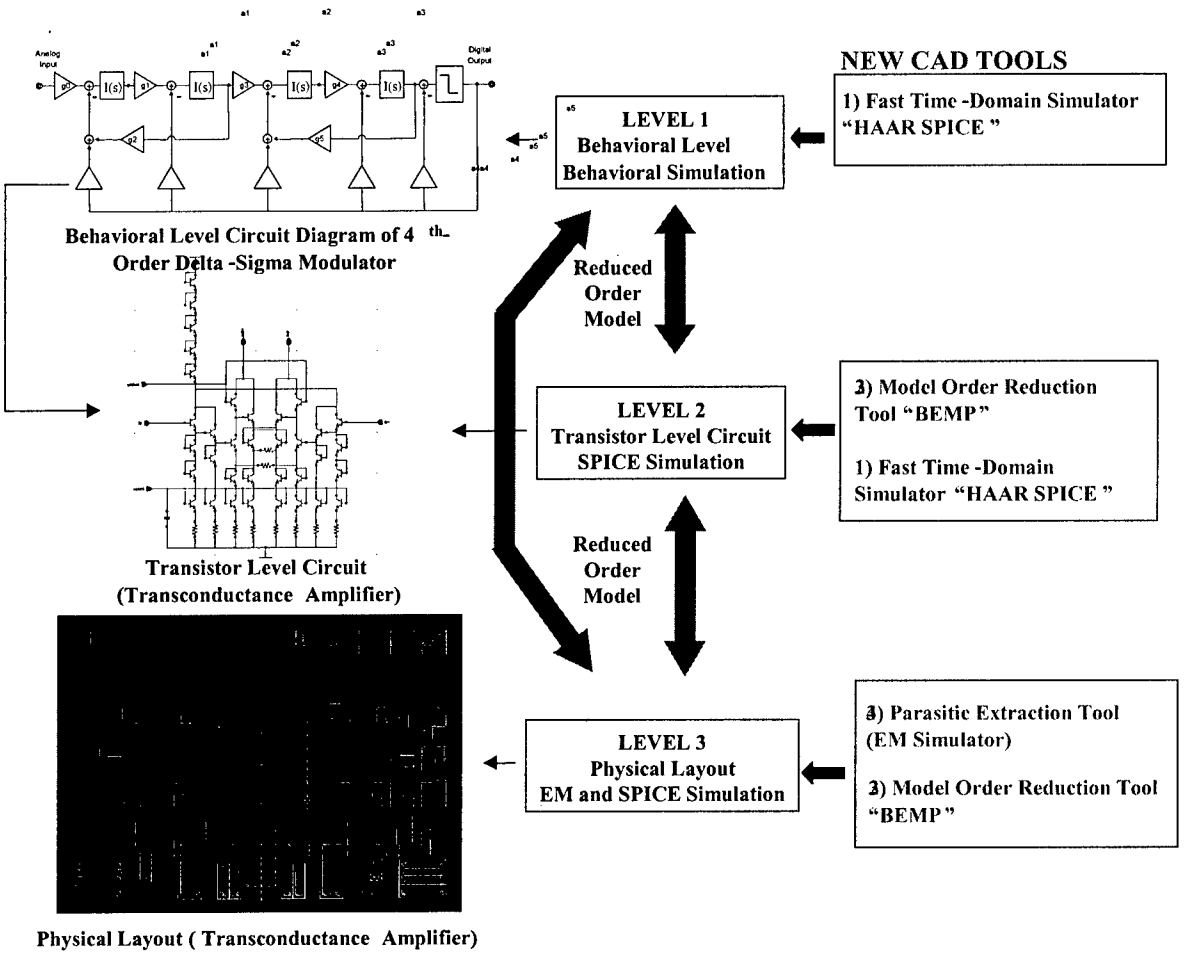


Fig. 1: HRL's New Mixed-Signal IC Design Flow (developed under DARPA NEOCAD program)

This program addressed the following areas in mixed-signal circuit design:

1.1.1 Fast Time-Domain Simulation

A fast time-domain simulation using a Jacobian-free algorithm was used to support both the behavioral and circuit level (Level 1&2) simulations. This new and innovative circuit simulation approach is applicable for both architectural and transistor-level time-domain circuit simulations. The heart of this method is to partition the circuits into linear and non-linear subblocks and lump the nonlinearities together in a form such that the need of calculating computationally expensive Jacobians is totally eliminated. Some of the salient features on this method are as follows:

- Works in the time domain, so the circuit nonlinearities can be easily included and treated
- Provides a uniform error distribution for the solution
- Is compatible with reduced order models
- Requires much less simulation time than SPICE calculations with the same accuracy

1.1.2 Model Order Reduction

Model order reduction (MOR) methods, such as asymptotic waveform evaluation, Pade-via-Lanczos and Arnoldi algorithms, provide macro-model representations of system and subsystem level circuit blocks in a form appropriate for network-oriented SPICE and behavioral system-level circuit simulations. The resultant macro-models can capture all the necessary characteristics of the original multi-port active and passive circuits with many fewer degrees of freedom by eliminating the internal nodes in the circuit. This traditional MOR method was extended to a more general Chebyshev- and wavelet-expansion-based MOR method, which is now part of Broadband efficient macromodeling program (BEMP).

BEMP provides a means for generating transfer functions of multi-port circuits. The transfer functions can be used to connect between levels in a design hierarchy and also to enable reduction in CPU time. Multi-variable optimization and automated synthesis methods can be developed around these transfer functions

1.1.3 Parasitic Extraction

We developed a fast, high-order, frequency-domain, full-wave electromagnetic simulation algorithm for effective parasitic extraction of mixed signal circuits directly from the layout. The method is based upon a high-order, method-of-moment solution scheme for the Mixed Potential Integral Equation (MPIE) and a fast iterative algorithm based on the high-order non-uniform FFT method. This innovative approach uniquely combines a quadrature-point-based discretization scheme for the MPIE and the high-order pre-corrected FFT method so that the algorithm scales as O(NlogN). This results in reduced Central Processing Unit (CPU) time and memory requirements compared to other frequency-domain full wave electromagnetic simulation techniques, such as the method of moments, the finite element method, and the method of lines. The initial parasitic extraction tool was able to analyze dense and complex passive printed circuits on a one-layer conductor-backed planar substrate. The outputs of the parasitic extraction tool were used to calculate wideband frequency-domain scattering parameters of the circuit, which are the input parameters to the MOR tools.

1.2 Overall Program Milestones

The goal for milestones are as listed for the Baseline Program for Baseline Program Schedule.

MILESTONE	Met (Yes/No)	Comments
Fast solution for circuit simulations	Yes	HAARSPICE-ELD algorithm shown to be 10X better than SPICE
Circuit equations in terms of wavelet bases	Yes	Developed connection coefficient operator matrices for all Daubechies class of wavelets
Adaptive wavelet expansion functions	Yes	Developed but not implemented in HAARSPICE
Circuit equations in terms of wavelet bases	Yes	Developed but not implemented in HAARSPICE
Automated prototype for wavelet-based time domain algorithm	Yes	Developed first time-domain DAE solver using Haar wavelets
MOR methods and algorithms	Yes	Developed and implemented in BEMP 1.0 release
Special MOR in time domain	Yes	Developed and implemented in BEMP 3.0 released
MOR for active components	Yes	Developed Verilog-AMS models of 4^{th}-order, 2^{nd}-order CT delata-sigma modulators
Automated prototype for MOR algorithms		
Model library development	Yes	Developed for HAARSPICE CADENCE-compatible NEOCADLib
Automated prototype for circuit and optimization algorithms	No	Currently HAARSPICE does not support optimization.
Fast time-domain circuit simulator validation	Yes	Demonstrated at Neocad Final Review Meeting (San Diego) HAARSPICE 2.0 and User Manual Released
Circuit synthesis and optimization tool validation	No	Currently HAARSPICE does not support synthesis.
Fast high-order, frequency-domain, full-wave EM simulation algorithm	Yes	QS-PCFFT algorithm developed
Automated prototype of EM simulation algorithm	Yes	QS-PCFFT validated and benchmarked
Final report	Yes	This Document

Table 1. NeoCAD Milestones

CHAPTER 2 FAST TIME DOMAIN MIXED-SIGNAL CIRCUIT SIMULATION

2.1 HAARSPICE Algorithms

We begin this chapter by describing the details of the algorithms used in HAARSPICE. Section 2.1 describes the mathematics behind circuit simulators and formulates the solver method used by the algorithms. Section 2.2 of this chapter then describes the software architecture of HAARSPICE and its implementation using the Object Oriented programming (OOP) methods. Finally, in section 2.3 we present benchmark results and list the performance metrics.

2.1.1 Mathematical Background

Circuits are represented as differential algebraic equations (DAEs) using modified nodal analysis technique (described in sec. 2.2.2). In the matrix form the DAEs can be as follows:

$$C\dot{x}(t) + Gx(t) = F(x(t)) + u(t) \qquad (1)$$

where $C = C_{M\times M}$ is the capacitance matrix used for representing magnetic and electric storage devices (i.e. inductors and capacitors), $G = G_{M\times M}$ is the transconductance matrix representing the lossy elements (i.e. resistors and transformers) in the circuit, $u(t)$ is the known input at a given node and $F = F_{M\times 1}$ is the non-linear operator matrix of M unknowns (i.e. $x(t)$). Integration of Eq. (1) gives,

$$C[x(t) - x(-\infty)] + G\int_{-\infty}^{t} dt'x(t') = \int_{-\infty}^{t} dt'F(x(t')) + \int_{-\infty}^{t} dt'u(t') \qquad (2)$$

The unknowns, $x(t)$, and the input, $u(t)$, can be expanded in N known functions, $\phi_i(t)$,

$$u(t) = \sum_{i=1}^{N} \phi_i(t)\hat{u}_i \qquad x(t) = \sum_{i=1}^{N} \phi_i(t)\hat{x}_i$$

where $\phi_i(t) \to 0$, as $|t| \to \infty$. Therefore for N time steps t_j,

$$\Phi_{ji} = \phi_i(t_j)$$

$$\Sigma_{ji} = \int_{-\infty}^{t_j} dt'\phi_i(t')$$

and the integral form can be written in terms of $\hat{x}_i, \phi_i(t)$ as

$$C\Phi_{ji}\hat{x}_i + G\Sigma_{ji}\hat{x}_i = \int_{-\infty}^{t} dt'F(t', x(t')) + \Sigma_{ji}\hat{u}_i \qquad (3)$$

Approximate $F(x(t)) = \sum_{i=1}^{N} \phi_i(t)\hat{f}_i$ and approximate the coefficients by sampling. Thus, $\hat{f}_i = \Phi_{ji}^{-1} F(t_j, \Phi_{ji}\hat{x}_i)$. The Eq. 3 can now be solved in discrete steps,

$$\{C\Phi_{ji} + G\Sigma_{ji}\}\hat{x}_i = \Sigma_{ji}\{\Phi_{ji}^{-1}F(t_j, \Phi_{ji}\hat{x}_i) + \hat{u}_i\} \quad (4)$$

The form described in Eq. 4 can be used to iteratively solve the linear equations for $\hat{x}_i$. Iterative solvers are appealing because they automatically use the sparseness of C and G. Though G is non-singular, it can be extremely ill-conditioned making iterative methods ineffective.

A non-iterative linear solver is obtained by multiplying Eq. (4) by G^{-1} and setting $A = G^{-1}C$

$$A\Phi_{ji}\hat{x}_i + \Sigma_{ji}\hat{x}_i = G^{-1}\Sigma_{ji}\{\Phi_{ji}^{-1}F(t_j, \Phi_{ji}\hat{x}_i) + \hat{u}_i\} \quad (5)$$

One can express A in the eigen-basis, $A = S\lambda S^{-1}$, with λ the diagonal matrix of the eigenvalues of A and S the eigenvectors. With $\hat{y}_i = S^{-1}\hat{x}_i$ and $B = S^{-1}G^{-1}$,

$$\{\lambda\Phi_{ji} + \Sigma_{ji}\}\hat{y}_i = B\Sigma_{ji}\{\Phi_{ji}^{-1}F(t_j, \Phi_{ji}S\hat{y}_i) + \hat{u}_i\} \quad (6)$$

For non-symmetric A, λ and S are complex.

2.1.2 Wavelet Expansions

The essential property of wavelets used to develop fast methods is the compact support of the basis, $\phi_i(t)$. Compact support makes the matrix Φ_{ij} sparse with non-zero elements close to the diagonal. The integral operator Σ_{ji} becomes close to an upper triangular matrix. The specifics of the wavelets enable one to make definite statements about Φ_{ij} and Σ_{ji}. A particular well-known property is that both Φ_{ij} and Σ_{ji} can be made Toeplitz with zero padding, enabling a fast FFT convolution operation of order *O(NlogN)* to evaluate their inverses. We begin with using the most basic type of wavelets called the Haar Basis wavelets in Eq. 6.

In the Haar Wavelets sampled at $t_j = (j - \frac{1}{2})N$, the measure of support is $\Delta = 1/N$. Therefore $\Phi_{ij} = \delta_{ji}/\Delta$ and $\Sigma_{ji} = \Theta(j - i)$ where

$$\Theta(n) = \begin{cases} 1 & n > 0 \\ 1/2 & n = 0 \\ 0 & n < 0 \end{cases}$$

In the Haar basis Eq. 6 becomes

$$\{\frac{\delta_{ji}}{\Delta}\sigma + \Theta(j - i)\}\hat{y}_i = B\Theta(j - i)\{\delta_{ik}F(S\hat{y}_i) + \hat{u}_i\} \quad (7)$$

The operation count needed to solve this equation can be estimated. Inversion of operators of the form $\alpha_m \delta_{ij} + \Theta(j-i)$ is an $O(N)$ operation. Inversion for all α_m is $O(MN)$. Application of S to all $\hat{y}_i$ is $O(M^2N)$. Evaluation of F is also $O(M^2N)$ though in general, F is very sparse in M. The application of B and the map $\hat{y}_i \rightarrow \hat{x}_i$ is $O(M^2N)$. Thus, the method scales approximately as $O(M^2N)$, which is the same scaling as implicit time-domain methods when a dense inverse of the system matrix is used.

2.1.3 Algorithm Steps

For each unknown $\{x(t)\}_m$ a unique set of wavelets can be chosen. Within the Haar wavelet space, the number of functions can be chosen dependent on m,N_m. If the circuit is characterized by a few high-frequency signals and many low frequency signals, selection of a wavelet basis specific to signal bandwidth leads to a considerable reduction in simulation time. Based on this principal we developed an algorithm that is 10x faster than the commercial SPICE solver for time-domain simulations of mixed-signal circuits.

Algorithm **: Eigenspace Linear Domain Solver (ELD)**

Step 1: Partition circuit into Analog and Digital sub-blocks
Assumption 1: All Digital Blocks are synchronous type.
Assumption 2: Time invariance between clock edges.

Step 2: Find DC operating point using homotopy methods
Homotopy methods used: arc length and Gmin stepping

Step 3: Solve Differential Algebraic Equation (i.e. Eq. 6)
$Y(t) = exp(h/\Lambda)*Y(t-h)$
Λ – *Eigenvalues*
h – Variable time step based on Error analogous to wavelet resolution

Note : Matrix-Vector Computations done on multiple Linux Clusters
gives 7-10x Speed up in comparison with conventional time stepping methods

The benchmark results of the above algorithm can be found in Section 2.3 of this chapter.

Other algorithms developed include:
ESE - Eigenspace Evolution Non Linear Domain Solver (Jacobian Free)
ENL - Eigenspace Non Linear Domain Solver

Details of both of the above mentioned algorithms are found in the HAARSPICE Manual reprinted in Appendix A of this report.

2.2 Implementation in HAARSPICE

We have used an Object Oriented Programming (OOP) approach in the development of HAARSPICE. OOP allows software developers to come up with classes that have data and functions encapsulated into self-contained entities. Besides data and function encapsulation, OOP provides other features, such as function overloading, inheritance and function hiding, making OOP a desirable tool in the development of circuit simulators. In this chapter we describe the HAARSPICE Programming Model, Matrix formulation and Command structure. For more details on HAARSPICE syntax and operation, the reader can refer to HAARSPICE User's Manual in Appendix A of this report.

2.2.1 HAARSPICE Programming Model

Fig. 2 shows the general framework of our simulator. In HAARSPICE a circuit is described using a text file containing a list of the required electronic components. The circuit netlist is obtained using a set of standalone commands in which each command performs a particular type of analysis. HAARSPICE commands will be explained later in this chapter.
Every line in the netlist is read as a "**StreamElement**". HAARSPICE parses the stream element into different classes, such as the "**Component**" class, the "**Model**" class and the "**Subcircuit**" class.

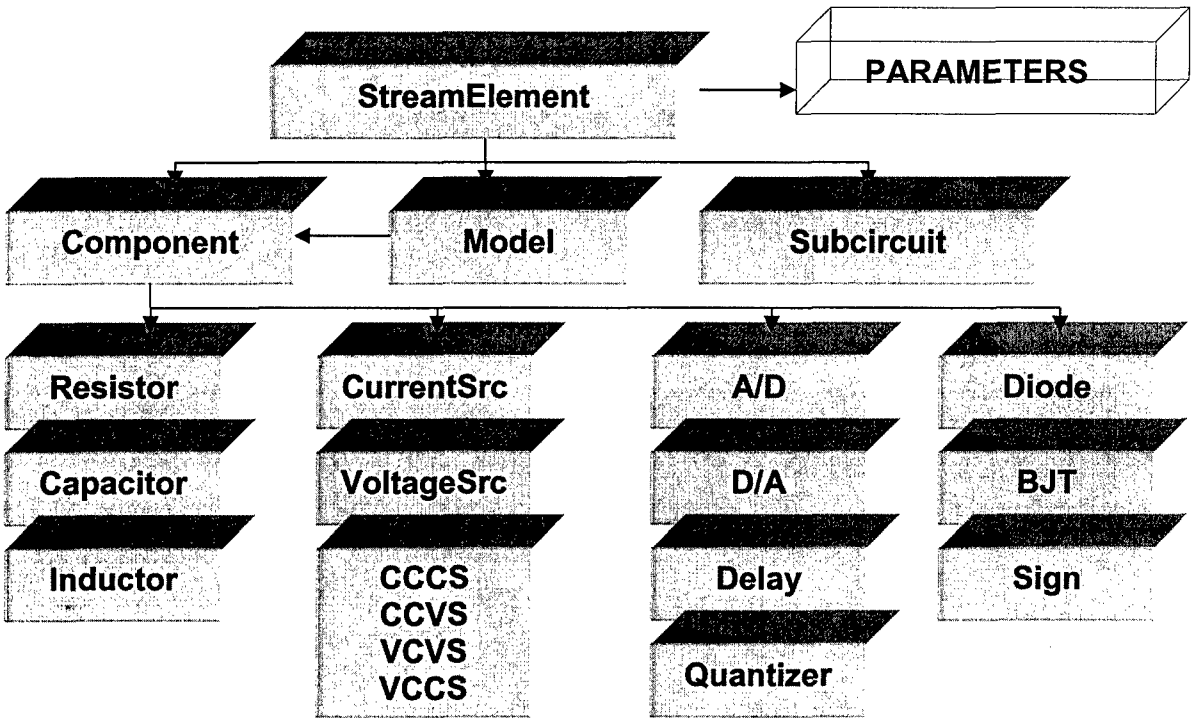


Fig. 2: HAARSPICE Programming Model

In the component class we have the non-linear and linear components used in our circuits. Fig. 2 shows all the elements in the component class. The OOP method used in HAARSPICE makes adding new classes or components relatively easy compared to SPICE3. Adding a new model in SPICE3 requires modifying over 30 C-files while in HAARSPICE adding a new model requires making only a few changes to the component class.

Below is the structure of the Diode Class used in HAARSPICE.

```
class Diode : public Component {
public:
    Diode() : _lambda(1) {}
    Diode(const StreamElement& def, class StringToIndex& stoi);
    virtual void F(RArray&, double, const RArray&) const;
    virtual void JacobianF(RMatrix&, double, const RArray&) const;
    virtual Component* create(const StreamElement& def, class
StringToIndex& stoi) const;
    virtual const char** cnames() const { return _cnames; }
    // returns true if limited to devices conductance.
    virtual bool setLeakConductance(double leakG_);
    virtual bool lambda(double _lambda_) { return (_lambda = min(1.0,
_lambda_)) == 1.0; }
    virtual Component& drop(int gni);
    virtual vector<int> giF() const;
private:
    DDT3 dQdt;
    bool drop_i;
    double Is, Vt, leakG;
    double vtm, bv, ibv;
    // charge based capacitance
    double tt, phi, cjo, m, f1, f2, f3, fcp;
    static const char* _cnames[];
    double _lambda;
};
```

The class Diode has two public functions F() and JacobianF() that are used to describe the nonlinear transfer function and its jacobian. The functions setLeakConductance and lambda are used for convergence and homotopy respectively. The private section lists all the other model parameters, such as saturation current (Is), threshold voltage (Vs), zero-bias junction capacitance (Cjo), transit time (tt).

Similar classes can be added for different devices. Shown below is another example of the BJT device class.

```
class BJT : public Component {
public:
      BJT() : IS(0) {}
      BJT(const StreamElement& def, class StringToIndex& stoi);
      virtual void F(RArray&, double, const RArray&) const;
      virtual void JacobianF(RMatrix&, double, const RArray&) const;
      virtual Component* create(const StreamElement& def, class
StringToIndex& stoi) const;
      virtual const char** cnames() const { return _cnames; }
      virtual vector<int> giF() const;
private:
      DDT3 dQdt;
      double IS, ISC, ISE, NC, NE, NF, NR, BR, BF;
      double VAF, VAR, IKF, IKR;
      double CJE,VJE,MJE,CJC,VJC,MJC,FC,TR,TF,XTF,ITF;
      double fbe1,fbe2,fbe3,fbc1,fbc2,fbc3;
      double fpbc,fpbe;
      double F1(double m , double f , double v)
      {
       return((v/(1 - m))*(1 - pow((1 - f), 1 - m)));
      }
      double F2(double m , double f)
      {
       return(pow((1 - f), (1 + m)));
      }
      double F3(double m , double f)
      {
       return(1 - f*(1 + m));
      }
      double Vt, gmin;
      static const char* _cnames[];
};
```

The DDT3 structure used in the BJT and diode class performs the third order time derivative. Also included are DDT and DDT2 for linear and 2nd order time derivatives. Having seen the structure of the models we now introduce the structure of the matrices that are solved by HAARSPICE.

2.2.2 Circuit Representation Matrices

Common matrix formulation methods for circuit analysis are State Variable, Modified Nodal Analysis (MNA) and the Tableau Method [3]. Inside HAARSPICE we have used the modified nodal matrix type of formulation. The general form of MNA for a Linear time Invariant (LTI) circuit is given as follows.

$$G\,v(t) \quad + \quad C\frac{d}{dt}v(t) = Mu(t), \quad v(0) = v_0$$

G = *all conductces and constants*

C = *all capacitor and inductor values*

M = *connection to the input source* $u(t)$

$v(t)$ = *vector of nodal voltages*

The advantage of MNA over conventional nodal matrix formulations using the Kirchoff Current Law (KCL) is that it can include independent voltage and dependent sources in the G matrix by adding a vector for branch currents. This helps in formulating a matrix that can be used to compute node voltages and branch currents concurrently. The Tableau method is similar but includes all the branch currents in the given circuit. The matrices generated using the Tableau method are large and sparse. In HAARSPICE, as in conventional SPICE3, we generate modified nodal matrices that are sparse and computationally manageable.

Below is a simple resistive circuit with its equivalent modified nodal matrix

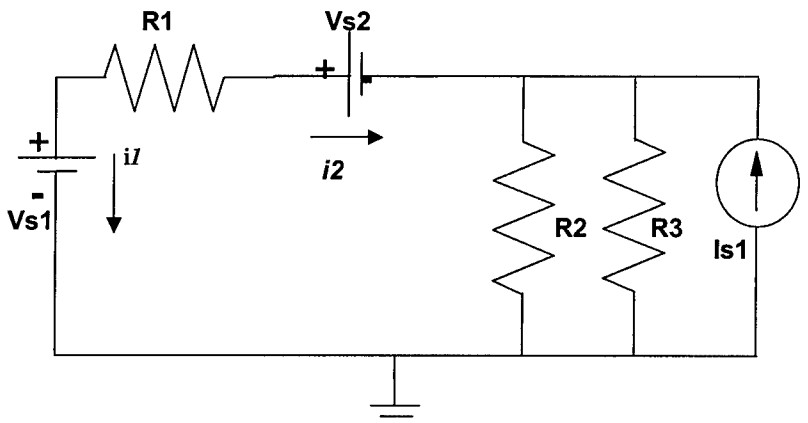


Fig. 3: A simple resistive circuit

$$\begin{bmatrix} G1 & -G1 & 0 & 1 & 0 \\ -G1 & G1 & 0 & 0 & 1 \\ 0 & 0 & (G2+G3) & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} v1 \\ v2 \\ v3 \\ i1 \\ i2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ I_{s1} \\ V_{s1} \\ V_{s2} \end{bmatrix}$$

The G-matrix shown above has two branch current entries for the two voltage sources. The datum or ground node is set to zero and is not included in the G-matrix formulation. The versatality of the MNA formulation is seen in its ability of matrix resuse for different kind of analyses. The modified nodal analysis matrix formulation for a simple RLC

network for time domain analysis is shown in Fig. 4. The inductors and capacitors are the storage elements and are described in the C-matrix (i.e. derivative matrix).

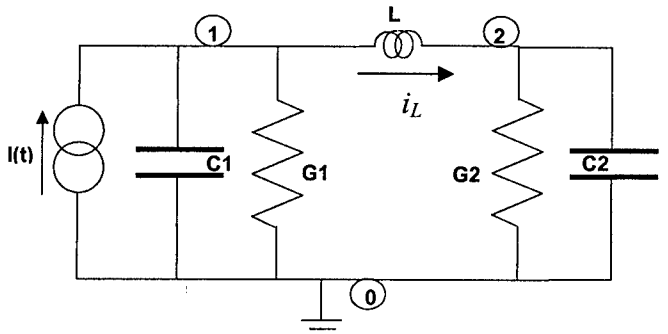


Fig. 4: A simple resistive circuit

$$\textit{SystemEquation} \quad G\,x + Cx' = w$$

$$G = \begin{bmatrix} G1 & 0 & 1 \\ 0 & G2 & -1 \\ 1 & -1 & 0 \end{bmatrix}, \quad C = \begin{bmatrix} C1 & 0 & 0 \\ 0 & C2 & 0 \\ 0 & 0 & L \end{bmatrix}, \quad w = \begin{bmatrix} i \\ 0 \\ 0 \end{bmatrix}, \quad x = \begin{bmatrix} v_1 \\ v_2 \\ i_L \end{bmatrix}$$

Inside HAARSPICE, the G and C matrices are stored in the Circuit class. Below is a section of the Circuit class:

```
class Circuit : public mtcode {
public:
    Circuit() : N(0), _xA(NULL), _yA(NULL), _BuA(NULL), _BFA(NULL),
    _pxT1A(NULL), _pyT1A(NULL), _heapA(NULL), _run(NULL) {}
    Circuit(int N_, string _name_)
    : N(N_), _name(_name_), _cnG(0), _cnS(0), _singularS(false),
    _gmin(0), _cmin(0), _reltol(1e-3),_abstol(1e-6),_iabstol(1e-12),
    _G(N_, N_, 0.0),_C(N_, N_, 0.0),_xA(NULL),_yA(NULL), _BuA(NULL),
    _BFA(NULL), _pxT1A(NULL),_pyT1A(NULL),_heapA(NULL),_run(NULL) {}
    Circuit(const Circuit& c) { *this = c; }
    virtual ~Circuit();
    Circuit& operator=(const Circuit&);

    operator int() const { return N; }
    string name() const { return _name; }
    virtual RMatrix G() const { return _G; }
    virtual RMatrix C() const { return _C; }
    // returns true if all equations are at limit
    virtual bool setLeakConductance(double leakG);
    // returns true if all components are 1
    virtual bool lambda(double lambda);
    // F is non-linear function
```

```
    virtual RArray F(double, const RArray&) const{return RArray(); }
    // returns dx/dt = C^(-1)(F(x) - G x)
    virtual RArray operator()(double t, const RArray& xA) const;
    // for delay effects, time-index, time-array, x-matrix, zero-time, xero-time x
    // thA and txM have values from times less than tA[0].
    virtual RArray F(int tn,const RArray& tA,const RMatrix& xM, double t0, const
RArray& x0A,
        const RArray& thA, const RMatrix& xhM) const;
    virtual RArray u(double) const { return RArray(); }
    // analytic jacobian
    virtual RMatrix JacobianF(double, const RArray&)const{return RMatrix(); }
    // numerical jacobian
    RMatrix JacobianF(double t, const RArray& x2A, const RArray& x1A) const;
. . . . . .
. . . . . .
.....
```

The G and C matrices are of the type RMatrix. The node voltages and branch currents computed by HAARSPICE are stored in RArray. Both these structures are as follows:

```
#ifndef RMATRIX_H
#define RMATRIX_H
#include <matrix.h>
typedef Matrix<double> RMatrix;
double norm(const RMatrix& M);
double infinity_norm(const RMatrix& M);
#endif
```

```
#ifndef RARRAY_H
#define RARRAY_H
#include <array.h>
typedef Array<double> RArray;
#endif
```

With the circuit described by the G and C matrices, one can now solve for the circuit response using the different algorithms. For solving linear circuits we recommend using the eigenspace linear domain solver (eld) inside HAARSPICE, which gives the best results in terms of speed and accuracy. The details of this solver are described in the earlier section.

2.2.3 Simulation of Nonlinear Circuits

In conventional circuit simulators, time domain analysis of nonlinear circuits is done by linearization of the nonlinear components during each time step. In HAARSPICE we use two types of algorithms: the first type uses conventional methods such as Newton-Raphson that require computing the Jacobian and the second uses non-Jacobian methods. Shown in Fig. 5 is the simple model of the 4^{th} order delta-sigma modulator with all linear components and the quantizer being the only non-linearity.

The nonlinearity in a circuit is stored in the Circuit::F structure as shown below

```
RArray Circuit::F(int tn, const RArray& tA, const RMatrix& xM, double t0, const
RArray& x0A,
    const RArray&, const RMatrix&) const {
```

```
    return tn >= 0 ? F(tA[tn], xM.row(tn)) : F(t0, x0A);
}
```

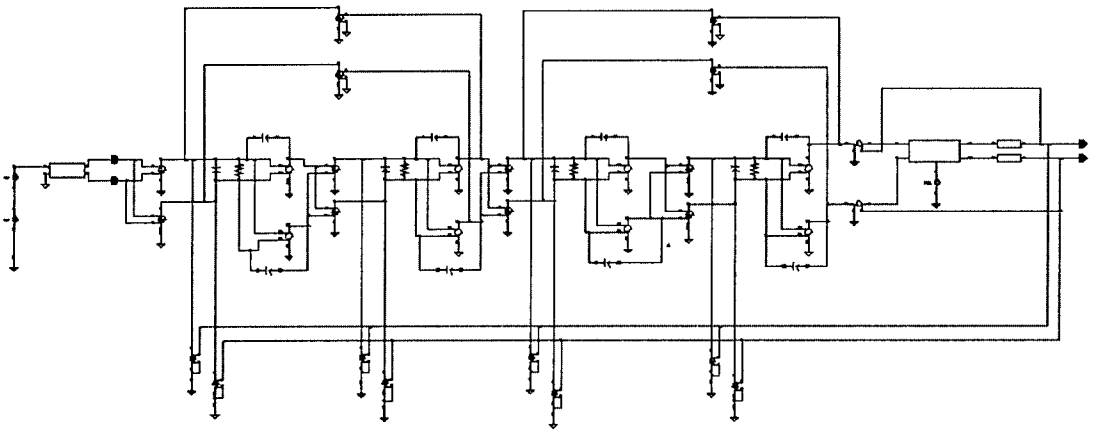


Fig. 5: 4th order delta-sigma modulator with quantizer nonlinearity

HAARSPICE-eld makes the RMatrix xM = elaS(circuit, N, runTime) call to solve this circuit. The elaS routine is executed by RMatrix ELASolver::operator()

The HAARSPICE-eld algorithm performs the time-domain simulation with 10X improvement is speed in comparison with SPICE. The FFT of the output from a 4th order delta-sigma modulator using a 16384 point Blackman window is shown in Fig. 6.

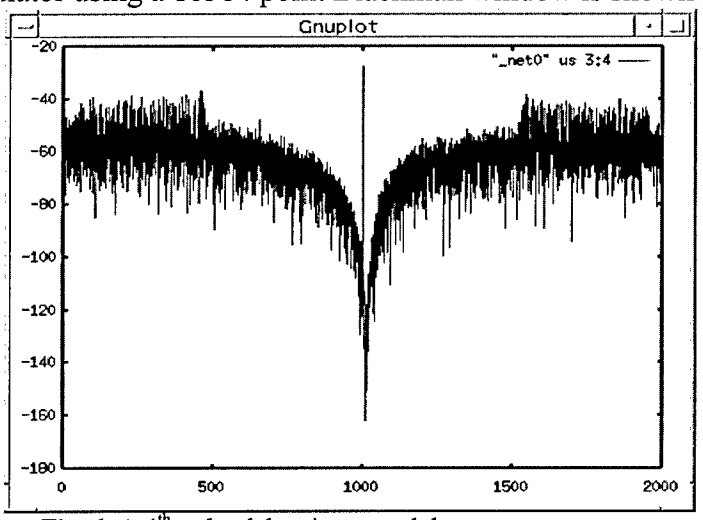


Fig. 6: A 4th order delta-sigma modulator output spectrum

2.2.4 HAARSPICE Commands

The HAARSPICE circuit simulator is a suite of programs used for time domain and DC analyses of linear and nonlinear analog circuits and systems. The HAARSPICE suite of programs runs on Linux 7.02 and has a text-based interface. Typing "**haarSPICE**" on the command line prompt yields a display of the following options:

enl - Eigenspace non-linear solver (jacobian)
eld - Eigenspace linear domain solver
ese - Eigenspace evolution nl-domain solver(jacobian free)
dcs - DC sweep
snlc- Supported netlist components
dcp - DC Point of circuit at t = 0

The current status of **HAARSPICE** supported components can be checked by typing **snlc** (i.e., supported netlist components), on the command line prompt.

available netlist components:

tag (...) atod|AtoD parameters ???
tag (...) bjt|NPN|PNP parameters ???
tag (nodeA nodeB) capacitor c=value "value of capacitance between nodeA and nodeB"
tag (...) cccs|cccs_hdl parameters ???
tag (...) ccvs parameters ???
tag (...) delay|delay_op parameters ???
tag (...) quantizer3|differentialQuantizer parameters ???
tag (...) diode|diode_hdl|diode_sch parameters ???
tag (...) dsmod|delta-sigma-modulator parameters ???
tag (...) dtoa|DtoA parameters ???
tag (nodeA nodeB) inductor l=value "value of inductance between nodeA and nodeB"
tag (...) isource|csource parameters ???
tag (...) quantizer1|quantizer parameters ???
tag (nodeA nodeB) resistor r=value "value of resistance between nodeA and nodeB"
tag (...) sign parameters ???
tag (nodeA1 nodeA2 nodeB1 nodeB2) transformer n1=value n2=value2 "value1/value2 is gain between nodes A and nodes B"
tag (...) vccs parameters ???
tag (...) vcvs parameters ???
tag (...) vsource parameters ???
Note : ??? denotes instance parameters

The command used to run the different HAARSPICE algorithms has a common structure. For example, the command used to execute the HAARSPICE-eld algorithm is given by

eld (filename|-testcircuit) log2(timesteps) run-time [-savenodes] [-writecircuit]

filename:

Currently, "HAARSPICE-eld" supports Cadence Analog Artist generated "spectre like" netlist. The components used are described in the earlier section. All spectre related analyses statements are ignored by HAARSPICE-eld.

Variables. Includes two sub-attributes:

timesteps order: <4-20> number of timesteps
runtime: time for which the simulation is run

savenodes. By default all the nodes in the netlist are saved. Also includes branch currents or pseudo-nodes that describe the currents in a Modified Nodal Matrix.

Writecircui. Writes out information about circuit

~/eld/order#,totaltime.
For example, a directory generated by running a simulation for 4.098 μs with an order of 15 (2^{15}) is named *"/eld/2E15,4u"*

The data files generated are in ascii text format and have 4 columns of numbers in double float type. An example of a data file

<time>	<value>	<frequency>	<spectral density /w Blackman window>
6.250000000000e-05	4.169160088445e-11	0.000000000000e+00	-5.365224910837e+01
1.875000000000e-04	5.000000005200e-01	2.441406250000e-01	-5.312466206937e+01
3.125000000000e-04	5.000000007146e-01	4.882812500000e-01	-5.500754703673e+01
.....	.....	.....	.....
.....	.....	.....	.....

The command for eigenspace evolution nonlinear domain solver (ese) inside HAARSPICE also has similar command structure with the addition of integration_order option.

ese (-netlist:filename) integration_order log2(timesteps)run-time [-savenodes] [-writecircuit]

The command_names[] structure stores the commands used with HAARSPICE. One can increase this structure to add new commands. The next section in this chapter lists the simulation and benchmarking results.

2.3 Simulation Results and Benchmarking

In this section we present some of the results that were demonstrated at the final DARPA Neocad review meeting in Coronado Bay on July 15-16, 2004.

Performance Metric

HAARSPICE-eld performance for a 4th order Band Pass Delta-Sigma Modulator circuit was tested on a DELL C800 Laptop running Linux (Red Hat 7.02) operating system.

The Table shown below gives the test results:

HAARSPICE -eld	SPICE/Spectre	Platform
63 seconds	593 seconds	Linux 7.02

In comparison with time-stepping algorithms used in SPICE, HAARSPICE-eld gives ~10X improvement in speed.

A performance metric giving the measure of accuracy is the **Dynamic Range** of the simulator. To measure the DR of the simulator we perform a two-tone test on the 4th order Delta-Sigma Modulator.

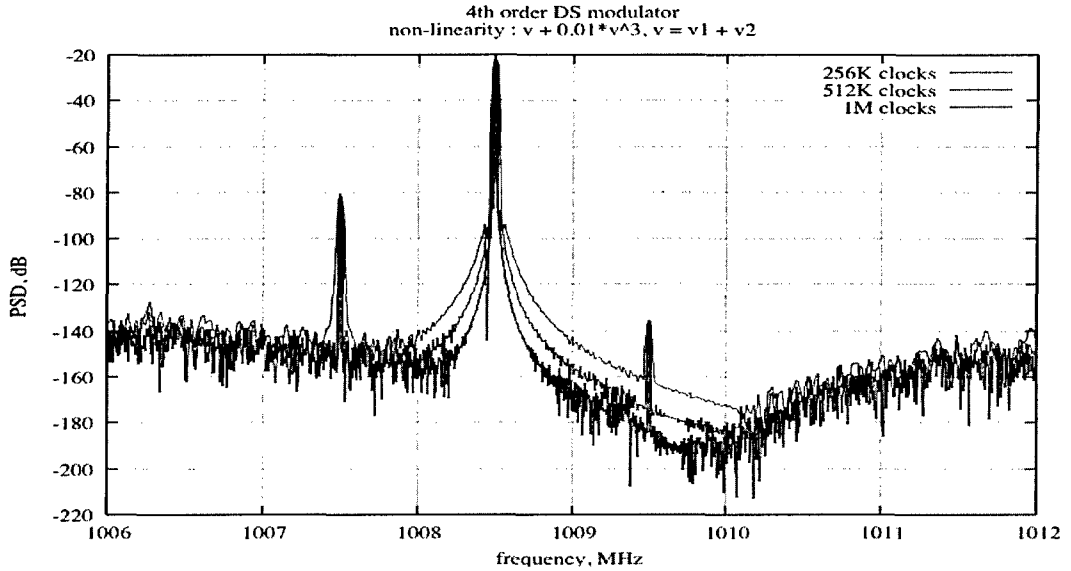


Fig. 7: Spectrum of the Two-Tone Test measured at the output for different number of clock cycles.

The Two Tone Test has one input signal u_1, at 1008.5 MHz with magnitude of 500mV and a second signal u_2, at 1007.5 MHz with a magnitude of 0.5mV. The nonlinearity introduced is through the 1st Trans-conductance Cell (i.e. Gm-Cell) of the Modulator. The only other nonlinearity is that of the quantizer. The 1st Gm-Cell nonlinearity generated a 3rd order harmonic is seen at 2x1008.5 – 1007.5 = 1009.5MHz as seen in Fig. 7.

Performance Metric

The Spurious Free Dynamic Range in the Two-Tone test is taken as the Dynamic Range of the simulator.

From Fig. 7, one can compute the DR to be Pmax - Pmin = -20dB - (-200) dB = 180 dB To resolve the 3rd order Harmonic and get a DR of more than 170 dB it took HAARSPICE-eld 1Million Clock Cycles.

The Table shown below gives the DR with two-tone test:

HAARSPICE (eld routine)	SPICE/Spectre	Clock Cycles
180 dB	173 dB	512 K

This makes HAARSPICE-eld particularly useful in circuits for communications, radar and other systems that require high Dynamic Range.

Performance Metric

HAARSPICE-ese performance for a 3-latch Quantizer used in a 4th-order Band-Pass Delta-Sigma Modulator circuit was tested on a DELL C800 Laptop running Linux (RedHat 7.02) operating system.

The Table shown below gives the test results.

HAARSPICE (ese)	HAARSPICE (enl) Generic Time-stepping code	Platform
400 seconds	> 5000 seconds	Linux 7.02

HAARSPICE-ese is not optimized. We are working on adding ARPACK routines to do mathematical computations. We expect the optimized code to give 10x speed up in comparison with uniform and adaptive time stepping code. Detailed simulation results can be found in the HAARSPICE manual in the appendix section.

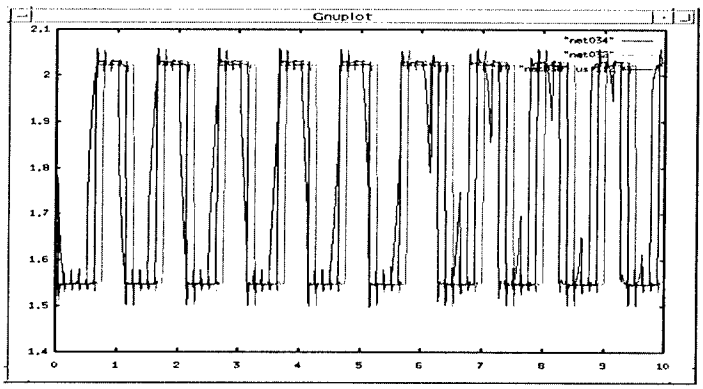


Fig. 2: 3-Latch Quantizer output

CHAPTER 3 MODEL ORDER REDUCTION TOOL (BEMP)

BEMP (Broadband Efficient Macro-modeling Program) was developed at the Georgia Institute of Technology to implement the research conducted on model order reduction as part of the NeoCAD program. The tool is coded using the C++ language and is executable on the Windows operating system. This chapter discusses the methodology used in BEMP to obtain model order reduction, the associated algorithms, and some benchmarking results.

BEMP integrates the electromagnetic behavior of passive systems into conventional computer-aided-design (CAD) tools through the generation of multi-port macro-models of the passive systems. A macro-model is a black-box representation of a system which accurately reproduces its port-to-port behavior as shown in Fig. 3.1. Using such macro-models, designers can account for the electromagnetic effects of passive systems during the design and simulation of multi-GHz electronic systems.

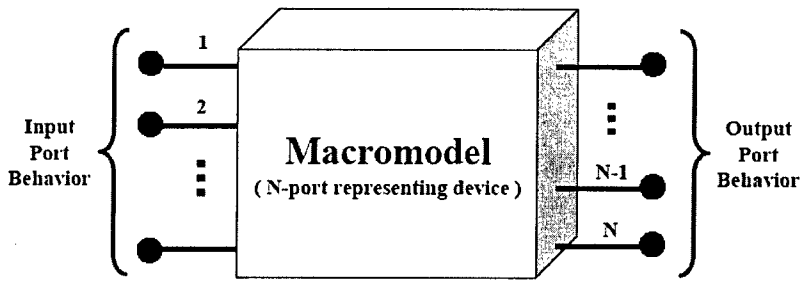


Fig. 3.1: Macro-model

3.1 Methodology used

Realistic distributed passive networks operating over a broad frequency range often contain hundreds of resonant peaks and the amplitude variation of the frequency response can be large. The representation of such systems using macro-models requires high order macro-models with hundreds of resonant poles. This can be a cumbersome task due to the large amplitude variation and the broad frequency range to be covered, and can lead to numerical problems. Hence, ensuring accuracy along with stability and passivity over a broad frequency range becomes a challenge. BEMP overcomes this challenge through the use of rational functions for approximating the frequency response of a passive network. The representation thus obtained is compensated for enforcing system characteristics like stability and passivity, while maintaining accuracy. Finally the compensated representation is converted into a lumped-parameter net-list that can be directly integrated into a circuit simulator like SPICE.

3.1.1 Rational Functions

Using rational functions, the frequency response $H(s)$ of any linear time-invariant passive network can be represented as:

$$H(s=j\omega)=\frac{\sum_{ns=0}^{NS} a_{ns}s^{ns}}{\sum_{ds=0}^{DS} b_{ds}s^{ds}} \tag{1}$$

where $s=j\omega$, ω is the angular frequency in radians per second, and a_{ns} and b_{ds} are unknown real numerator and denominator coefficients, respectively [1]. In (1), $H(s)$ can be scattering, admittance, or impedance parameters generated from an electromagnetic simulation or measurements. The goal of solving (1) is to represent the frequency response $H(s)$ using a rational function by computing the orders of the numerator and denominator, NS and DS, respectively, and the real vectors $[a]$ and $[b]$ consisting of the numerator and denominator coefficients, respectively.

Using filter theory, $H(s)$ in (1) can also be represented as a summation of filters in the form:

$$\begin{aligned} H(s)=&\sum_{m} H_{LP}^{m}(s)\\ &+\sum_{n} H_{BP}^{n}(s)\\ &+\sum_{k} H_{HP}^{k}(s)\\ &+\sum_{r} H_{AP}^{r}(s) \end{aligned} \tag{2}$$

where H^{m}_{LP}, H^{n}_{BP}, H^{k}_{HP} and H^{r}_{AP} are low pass, band pass, high pass, and all pass filters, respectively. The filters shown in (2) represent a complete set for representing any transfer function. The rational function in (1) can be rewritten in a pole-residue form as:

$$H(s)=\sum_{ds}\frac{r_{ds}}{s-p_{ds}} \tag{3}$$

where p_{ds} and r_{ds} are the poles and residues of the rational function, respectively, and ds is the order of the rational function. Combining (2) and (3) results in:

$$\begin{aligned} H(s)=&\sum_{m=1}^{LPN}\frac{r_m}{s-p_{mr}}\\ &+\sum_{n=1}^{BPN}\left(\frac{\alpha_n+j\beta_n}{s-p_{nr}-jp_{ni}}+\frac{\alpha_n-j\beta_n}{s-p_{nr}+jp_{ni}}\right)\\ &+\sum_{k=1}^{HPN}\frac{\psi_k s}{s-p_{kr}}\\ &+\delta+\eta s \end{aligned} \tag{4}$$

where p_{mr}, r_m, p_{nr}, p_{ni}, α_n, β_n, p_{kr}, ψ_k, δ, η, are real and the poles are simple or non-overlapping poles. By combining the complex conjugate poles, (4) can be rewritten as:

$$H(s) = \sum_{m=1}^{LPN} \frac{r_m}{s - p_{mr}} + \sum_{n=1}^{BPN} \frac{2\alpha_n(s - p_{nr}) - 2\beta_n p_{ni}}{(s - p_{nr})^2 + p_{ni}^2} + \sum_{k=1}^{HPN} \frac{\psi_k s}{s - p_{kr}} + \delta + \eta s \quad (5)$$

where the superscripts *LPN*, *BPN*, and *HPN* are the number of low pass, band pass, and high pass filters, respectively. In (2), the low pass filter $H^m{}_{LP}(s)$ consists of a real pole p_{mr} with residue r_m; the band pass filter $H^n{}_{BP}(s)$ consists of complex conjugate poles $p_{nr} \pm jp_{ni}$ with residues $\alpha_n \pm \beta_n$; the high pass filter $H^k{}_{HP}(s)$ consists of a real pole p_{kr} with residue ψ_k; and the all pass filter $H^r{}_{AP}(s)$ consists of residues δ and η.

3.1.2 Stability and Passivity

Macro-models constructed using rational functions need to satisfy the stability and passivity conditions for a linear time invariant passive system. The stability condition requires that for a stable system, the output response be bounded for a bounded input excitation. Hence, the rational function representing a stable system has to satisfy the following stability constraints:

1) The poles lie on the left half of the s-plane, implying that $p_{mr} \leq 0$, $p_{nr} \leq 0$, and $p_{kr} \leq 0$ in the pole residue form of the rational function.
2) The rational function does not contain multiple poles along the imaginary axis of the s-plane.
3) The difference between the numerator and denominator orders of the rational function does not exceed unity, implying that $|NS - DS| \leq 1$.

Among the above constraints, the second and third constraints are enforced by construction while the first constraint is satisfied in BEMP using a search procedure.

The passivity condition requires that a passive circuit does not create energy. Since non-passive macro-models combined with a stable circuit can generate an unstable time-domain response, this condition becomes important when the macro-models need to be combined with a larger circuit for time-domain simulation. Unlike the stability condition, it is more difficult to satisfy the passivity condition during the construction of macro-models. The passivity conditions for a multi-port network $[G(s = \sigma + j\omega)]$ are twofold, namely, 1) $[G(s^*)] = [G^*(s)]$ for all s, where * is the complex conjugate operator and 2) $[G(s)]$ is a positive real matrix, i.e., the product $z^{*T}[G^T(s^*) + G(s)]z$, for all s with $Re(s) > 0$ and any arbitrary vector z. When the rational function matrix $[H(s)]$ is used, these conditions are translated into the following passivity constraints [3]:

1) $[H(s)]$ does not contain poles on the right half of the s-plane.
2) $[H(s)]$ does not have multiple poles on the imaginary axis of the s-plane.
3) The coefficients of $[H(s)]$ are all real.
4) The real part of $[H(s)]$ must be positive semi-definite for all frequencies, implying that the eigenvalues of $Re\{[H(s)]\}$ are positive or zero for all frequencies.

The last constraint is derived from the maximum modulus theorem. Among the above constraints, the first and second constraints are included as part of the stability condition and the third constraint is satisfied by construction. The fourth constraint has been enforced analytically in BEMP the details of which are described below.

Based on the maximum modulus theorem, the passivity condition for a one-port network can be written as:

$$Re\{H(s = j\omega)\} \geq 0 \quad \forall\omega \tag{6}$$

It is important to note that $s = j\omega$ in (6) (and not $s = \sigma + j\omega$) simplifies the derivation of the analytical formulae for satisfying passivity of the macro-models. The basic idea behind the construction of passive macro-models for a passive system is that the summation of passive sub-networks is passive. The rational function $H(s)$ in (5) can be regarded as a summation of passive sub-networks consisting of complex conjugate poles and real poles with corresponding residues, δ and η. If every sub-network in (5) satisfies the passivity condition, the rational function $H(s)$ satisfies the passivity condition as well. Substituting $s = j\omega$ into (5), the rational function $H(s = j\omega)$ can be separated into the real and imaginary parts as:

$$H(s = j\omega) = H_R(j\omega) + jH_I(j\omega) \tag{7}$$

By regrouping terms, the real $H_R(s = j\omega)$ and imaginary H_I $(s = j\omega)$ parts of the rational function are shown in (8) and (9), respectively.

$$\begin{aligned} H_R(j\omega) = & \sum_{m=1}^{LPN} \frac{-\gamma_m p_{mr}}{p_{mr}^2 + \omega^2} \\ & + \sum_{n=1}^{BPN} \frac{2\omega^2(-\alpha_n p_{nr} + \beta_n p_{ni})}{(p_{nr}^2 + p_{ni}^2 - \omega^2)^2 + (2p_{nr}\omega)^2} \\ & + \sum_{n=1}^{BPN} \frac{2(p_{nr}^2 + p_{ni}^2)(-\alpha_n p_{nr} - \beta_n p_{ni})}{(p_{nr}^2 + p_{ni}^2 - \omega^2)^2 + (2p_{nr}\omega)^2} \\ & + \sum_{k=1}^{HPN} \frac{\psi_k \omega^2}{p_{kr}^2 + \omega^2} \\ & + \delta \end{aligned} \tag{8}$$

$$\begin{aligned} H_I(j\omega) = & \omega \sum_{m=1}^{LPN} \frac{-\gamma_m}{p_{mr}^2 + \omega^2} \\ & + \omega \sum_{n=1}^{BPN} \frac{2\alpha_n(-p_{nr}^2 + p_{ni}^2 - \omega^2)}{(p_{nr}^2 + p_{ni}^2 - \omega^2)^2 + (2p_{nr}\omega)^2} \\ & - 4\omega \sum_{n=1}^{BPN} \frac{\beta_n p_{ni} p_{nr}}{(p_{nr}^2 + p_{ni}^2 - \omega^2)^2 + (2p_{nr}\omega)^2} \\ & - \omega \sum_{k=1}^{HPN} \frac{\psi_k p_k}{p_{kr}^2 + \omega^2} \\ & + \omega\eta \end{aligned} \tag{9}$$

The passivity of each sub-network in (5) can be satisfied using the following formulae.

$$\begin{aligned} \gamma_m &\geq 0 \\ -\alpha_n p_{nr} \pm \beta_n p_{ni} &\geq 0 \\ \psi_k &\geq 0 \\ \delta &\geq 0 \end{aligned} \tag{10}$$

In (10), the analytically derived passivity formulae can be obtained from (8) by satisfying (6). Equation (5) can be generalized for a distributed multi-port network containing common poles as:

$$\begin{aligned} [H(s)] = &\sum_{m=1}^{LPN} \frac{[\gamma_m]}{s - p_{mr}} \\ &+ \sum_{n=1}^{BPN} \frac{2[\alpha_n](s - p_{nr}) - 2[\beta_n]p_{ni}}{(s - p_{nr})^2 + p_{ni}^2} \\ &+ \sum_{k=1}^{HPN} \frac{[\psi_k]s}{s - p_{kr}} \\ &+ [\delta] + [\eta]s \end{aligned} \tag{11}$$

where the residue matrices $[\alpha_n]$, $[\beta_n]$, $[\gamma_m]$, $[\psi_k]$, $[\delta]$, $[\eta]$ are a PxP matrix for a P-port network. In (11), $[H(s)]$ is a rational function matrix, which has to be positive semi-definite at all frequencies. Using the property of positive semi-definiteness, the passivity formulae in (10) for a multi-port network can be rewritten as:

$$\begin{aligned} eigenvalues\ of\ [\gamma_m] &\geq 0 \\ eigenvalues\ of\ [-\alpha_n p_{nr} \pm \beta_n p_{ni}] &\geq 0 \\ eigenvalues\ of\ [\psi_k] &\geq 0 \\ eigenvalues\ of\ [\delta] &\geq 0 \end{aligned} \tag{12}$$

The following properties of multi-port passivity formulae in (12) are apparent during the construction of the passive macromodel. These properties have been used for the construction of broadband passive macro-models in later sections.

1) The multi-port passivity formulae only depend on the poles and residue matrices, which are independent of frequency. Hence, the passivity of the circuit is satisfied over infinite frequency.
2) The multi-port passivity formulae are only enforced on each sub-network of $[H(s)]$, and there is no relationship for passivity between sub-networks except that they contribute to the overall response of the macromodel. This makes the method simple to use.
3) For compensating negative eigenvalues in (12), there are two free matrix variables $[\alpha_n]$ and $[\beta_n]$ related to two free variables of complex conjugate poles $p_{nr} \pm jp_{ni}$, a free matrix variable $[\gamma_m]$ related to a real pole p_{mr}, a free matrix variable $[\psi_k]$ related to a real pole p_{kr}, and a free matrix variable $[\delta]$. These can be suitably changed.
4) There are no constraints enforced on the residue matrix $[\eta]$.

The non-passive macromodel that does not satisfy the passivity condition needs to be appropriately modified so that the macromodel becomes passive. For enforcing passivity on generated macro-model in BEMP, fixed common poles and symmetric residue matrices are assumed during compensation. If negative eigenvalues are obtained in the residue matrices $[\gamma_m]$, $[\psi_k]$, $[\delta]$ in (12), negative eigenvalues are set equal to zero or

changed to a small positive value and then a new residue matrix is reconstructed. If the passivity formulae for complex conjugate poles with two residue matrix variables $[\alpha]$ and $[\beta]$ are violated, the negative eigenvalues of $[\alpha]$ are set equal to zero or changed to a small positive value and then a new residue matrix $[\alpha]$ is reconstructed. Based on the compensated matrix $[\alpha]$, the residue matrix $[\beta]$ is iteratively found for satisfying the passivity formulae in (12). A small positive value is used to ensure that the macromodel does not violate the passivity condition even though this may cause small numerical errors in the solution.

3.1.3 Least Squares Approximation

As mentioned earlier, (1) needs to be solved to compute the orders, *NS* and *DS*, and the real coefficient vectors, $[a]$ and $[b]$. Equation (1) can be rewritten in the form:

$$\sum_{ns=0}^{NS} a_{ns} s^{ns} - H(s) \sum_{ds=0}^{DS} b_{ds} s^{ds} = 0 \tag{13}$$

For a given $H(s)$ from either measured or simulated data, which represents the frequency response of a one-port network, (13) can be written as a matrix equation in the form [3]

$$[A] \begin{bmatrix} a \\ b \end{bmatrix} = [0] \Longleftrightarrow [A][x] = [0] \tag{14}$$

where the matrix $[A]$ is given by

$$[A] = \begin{bmatrix} re\{\sum_{ns=0}^{NS} s_1^{ns}\} & -re\{H(s_1) \sum_{ds=0}^{DS} s_1^{ds}\} \\ im\{\sum_{ns=0}^{NS} s_1^{ns}\} & -im\{H(s_1) \sum_{ds=0}^{DS} s_1^{ds}\} \\ \vdots & \vdots \\ re\{\sum_{ns=0}^{NS} s_k^{ns}\} & -re\{H(s_k) \sum_{ds=0}^{DS} s_k^{ds}\} \\ im\{\sum_{ns=0}^{NS} s_k^{ns}\} & -im\{H(s_k) \sum_{ds=0}^{DS} s_k^{ds}\} \end{bmatrix} \tag{15}$$

The vectors $[a]$ and $[b]$ in (14) are real coefficient vectors of the numerator and denominator, respectively. After pre-multiplying (14) with the transpose of $[A]$, (14) becomes

$$[A]^T[A][x] = [0] \tag{16}$$

which can be written as an eigenvalue equation

$$[A]^T[A][x] = \lambda_{min}[x] \tag{17}$$

where λ_{min} is the minimum eigenvalue of the matrix $[A]^T[A]$, where T is the transpose operator. From (17), the computation of the real coefficient vector $[x]$ requires the estimation of the integer orders, *NS* and *DS*. The orders in (1) can be estimated using the minimum eigenvalue tracking method shown in Fig. 3.2, which scans the orders *NS* and *DS* over a pre-determined range. In Fig. 3.2, the minimum eigenvalue has been plotted as a function of the orders, *NS* and *DS*. Based on (17), a non-trivial solution exists when $\lambda_{min} \approx 0$, as shown in Fig. 3.2. Hence, λ_{min} has been used as a parameter for tracking the optimum solution. In Fig. 3.2, the solution area represents the region with a valid solution, which corresponds to $\lambda_{min} = 10^{-14}$.

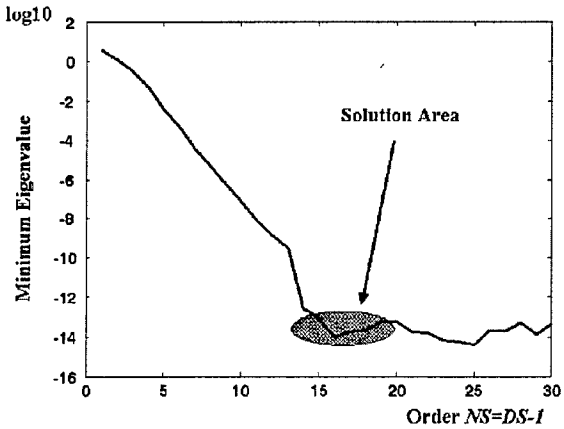


Fig. 3.2: Minimum eigenvalue versus the order NS = DS-1

From Fig. 3.2, for the specific example, the orders, $NS = 16$ and $DS = 17$, result in the eigenvector $[x]$ corresponding to the minimum eigenvalue. The eigenvector $[x]$ contains the coefficient vectors $[a]$ and $[b]$ of the rational function. The stability can now be enforced on the denominator coefficient vector $[b]$ of the eigenvector $[x]$ in (17) by applying the constraints $p_{mr} \leq 0$, $p_{nr} \leq 0$, and $p_{kr} \leq 0$. BEMP uses a root finding method for computing the poles and discards any unstable poles obtained in the process. For computing the residues corresponding to the stable poles, (18) is solved using the eigenvalue method discussed earlier. The matrix structure in (18) depends on the application and can be arbitrary. For a multi-port network containing common poles, the residues for each port are constructed independently by solving (18). Finally, the passivity formulae in (12) are used to enforce passivity of the rational function.

$$\begin{bmatrix} L_1^R & B_1^R & I_1^R & H_1^R & 1 & 0 & -G_1^R \\ L_1^I & B_1^I & I_1^I & H_1^I & 0 & \omega_1 & -G_1^R \\ & & & \vdots & & & \\ & & & \vdots & & & \\ L_t^R & B_t^R & I_t^R & H_t^R & 1 & 0 & -G_t^R \\ L_t^I & B_t^I & I_t^I & H_t^I & 0 & \omega_t & -G_t^I \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \vdots \\ \alpha_1 \\ \vdots \\ \beta_1 \\ \vdots \\ \psi_1 \\ \vdots \\ \delta \\ \eta \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ 0 \end{bmatrix} \tag{18}$$

where

$$L_t^R = real\ of \sum_{m=1}^{LPN} \frac{1}{j\omega_t - p_{mr}}$$

$$L_t^I = imaginary\ of \sum_{m=1}^{LPN} \frac{1}{j\omega_t - p_{mr}}$$

$$B_t^R = real\ of \sum_{n=1}^{BPN} \frac{2(j\omega_t - p_{nr})}{(j\omega_t - p_{nr})^2 + p_{ni}^2}$$

$$B_t^I = imaginary\ of \sum_{n=1}^{BPN} \frac{2(j\omega_t - p_{nr})}{(j\omega_t - p_{nr})^2 + p_{ni}^2}$$

$$I_t^R = real\ of \sum_{n=1}^{BPN} \frac{-2p_{ni}}{(j\omega_t - p_{nr})^2 + p_{ni}^2}$$

$$I_t^I = imaginary\ of \sum_{n=1}^{BPN} \frac{-2p_{ni}}{(j\omega_t - p_{nr})^2 + p_{ni}^2}$$

$$H_t^R = real\ of \sum_{k=1}^{HPN} \frac{j\omega_t}{j\omega_t - p_{kr}}$$

$$H_t^I = imaginary\ of \sum_{k=1}^{HPN} \frac{j\omega_t}{j\omega_t - p_{kr}}$$

$G_t^R = real\ of\ H(j\omega_t);\ frequency\ data$

$G_t^I = imaginary\ of\ H(j\omega_t);\ frequency\ data$

3.1.4 Broadband Macro-modeling

Realistic distributed networks such as interconnects operating over a broad frequency range often contain a large number of poles and the amplitude variation of the frequency response could be large. This can create numerical problems in (18) since the matrix $[A]^T [A]$ can become an ill-conditioned matrix. This is apparent in (1) where the power series expansion can have a large dynamic range. For instance, if the frequency response ranging from DC to 2 GHz needs to be approximated using 20 complex conjugate poles and 2 real poles, the dynamic range of elements in the matrix $[A]$ is from 1 to $(2\pi \text{ x } 2 \text{ x } 10^9)^{22}$, which causes the matrix to become ill-conditioned. This problem can be improved using frequency scaling and the computed poles and residues can be reconstructed using the scaling factor ω_0. Equation (13) can be written in the form

$$\sum_{ns=0}^{NS} a_{ns}\left(\frac{s}{\omega_0}\right)^{ns} - H(s)\sum_{ds=0}^{DS} b_{ds}\left(\frac{s}{\omega_0}\right)^{ds} = 0 \tag{19}$$

However, it has been shown that the scaling factor in (19) for approximating the frequency response does not result in significant improvement in the approximation beyond 20-30 poles. BEMP alleviates this ill-conditioned matrix problem by dividing the system frequency response into various frequency subbands and then processing each subband individually. As shown in (11), the rational function matrix $[H(s)]$ is represented in the pole-residue form as a summation of sub-networks. Both the stability and passivity conditions in each sub-network are satisfied using stability constraints and multi-port passivity formulae described earlier. Since stability constraints and passivity formulae are only enforced on each sub-network, there is no relationship for the stability and passivity conditions between sub-networks except that they contribute to the overall response of the macro-models. This enables the entire frequency response to be divided into sub-frequency bands (or subbands). The basic idea for the construction of broadband macro-models is that if complex conjugate poles and real poles can be extracted from a localized region of the frequency response, then the original frequency response can be divided into subbands, as shown in Fig.s 3.3(a) and (b).

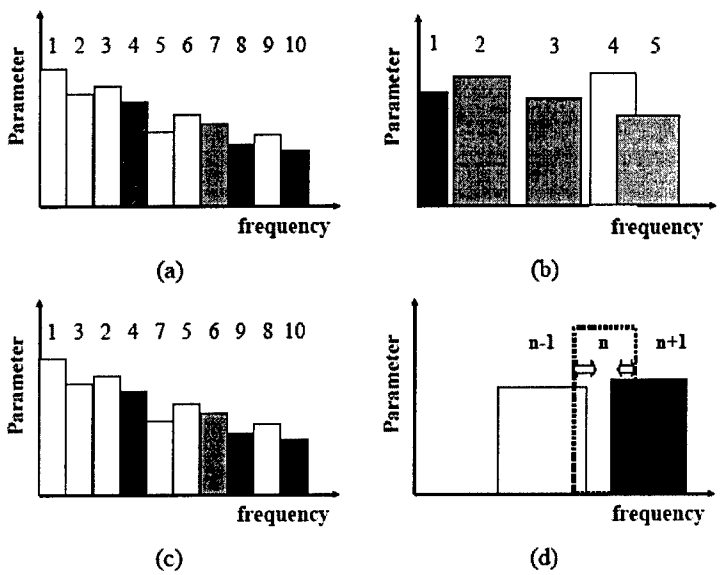


Fig. 3.3: (a), (b) Band division, (c) Subband reordering, and (d) Subband dilation

In Fig.s 3.3(a) and (b), poles from the localized region within a subband can be extracted. The process of dividing the entire frequency band of interest into smaller subbands is called as band division. The subbands are either of uniform or non-uniform width and contain a set of sampled frequency data, as shown in the Fig.. The criterion for choosing the width of each subband depends on the nature of the frequency response, but it is desirable to divide the entire frequency band into subbands where resonant peaks exist. In areas where no resonant peaks exist, the subbands are overlapped to minimize the number of subbands required. The orders *NS* and *DS* for each subband can be estimated using the minimum eigenvalue tracking method or the number of resonant peaks. Using the band division method, the macro-models for each subband can be extracted in parallel. However, macro-models from each subband can interact with each other since the extrapolation of each macromodel outside the subband of interest can produce a non-zero frequency response. This interaction can result in an erroneous frequency response when the macro-models are combined. During the least squares approximation process discussed earlier, poles located outside the subband can be frequently found since there are no constraints that limit the position of the poles. After removing the unstable poles, stable poles located outside the subband have to be handled since they amplify the interactions between subbands. Although these poles increase the accuracy in the subband of interest, they usually reduce the accuracy in adjacent frequency bands. Therefore, selectors are used to restrict the poles within the frequency band of interest during the generation of macro-models in each subband. This ensures smooth extrapolation and the absence of resonant peaks in adjacent bands. BEMP implements four types of selectors, namely, low pass, band pass, high pass and band reject selectors. To compensate for the inaccuracy in the subbands after the removal of poles outside the subbands, three methods namely, subband reordering, subband dilation and pole replacement methods have been implemented in BEMP.

In subband reordering method, the macro-models for each subband are constructed after subtracting the frequency response of the previous macro-models from the original frequency response. As discussed earlier, selectors are applied to handle the poles located inside or outside the frequency band. At every stage of this process, the subbands are reordered based on the magnitude of the frequency response in decreasing order. The macromodel for the subband with the largest magnitude is always constructed first, prior to the other subbands. This is shown in Fig. 3.3(c), where subbands 2 and 3 in Fig. 3.3(a) are interchanged during the macromodel construction process. Subband reordering reduces numerical errors caused by dominant poles in an adjacent subband.

In subband dilation method, the subbands are dilated to provide local correction in the region between adjacent subbands. This method results in an overlap between subbands, as shown in Fig. 3.3(d). In the Fig., subband 'n' is dilated to overlap subbands 'n-1' and 'n+1'. The amount of overlap is determined by the position of poles in each subband. If the poles are located at the boundary between bands, then these poles have the maximum effect on both frequency bands. As the poles are located farther away from the boundary, the effect of these poles on adjacent bands is minimized. Using this criterion, the subbands are suitably dilated such that the poles at the boundary lie in the overlap region. The macro-models of three subbands are then iteratively corrected by monitoring the error in three subbands using the pole replacement method discussed later. Though subband reordering and subband dilation methods minimize the interaction

between subbands, correction may once again be necessary since these methods may sometimes miss the poles at the boundary between subbands or generate spurious poles within the subband.

BEMP uses a pole replacement method to improve the accuracy of the constructed macro-models by discarding spurious poles and extracting accurate poles, as shown in Fig. 3.4.

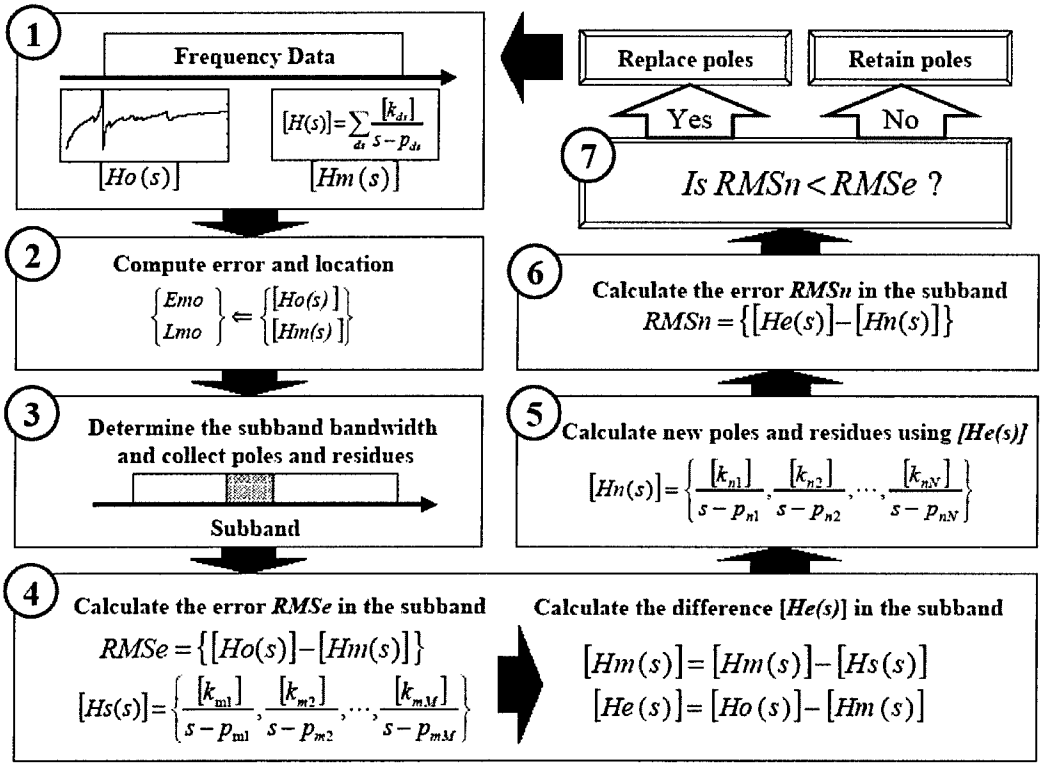


Fig. 3.4: Pole Replacement

The pole replacement method is applied after the macro-models for each subband are constructed. The details of the pole replacement method are described below:

1) The process begins with the comparison between the frequency response [*Hm*(*s*)] from the macro-models and the original frequency response [*Ho*(*s*)].
2) Based on the error criterion, the maximum difference value *Emo* and location *Lmo* between [*Ho*(*s*)] and [*Hm*(*s*)] are calculated and stored.
3) The algorithm searches for a set of poles around the location *Lmo* and determines the width of the subband to be recalculated.
4) The root-mean-square error *RMSe* between [*Ho*(*s*)] and [*Hm*(*s*)] in the subband is calculated and stored. The frequency response associated with the poles in the subband is then subtracted from [*Hm*(*s*)] in order to recalculate the frequency response of the subband. Then, the difference between [*Ho*(*s*)] and [*Hm*(*s*)] is calculated and stored in [*He*(*s*)].
5) Using [*He*(*s*)], new poles and residues are recalculated in the subband.
6) The frequency response using new poles and residues [*Hn*(*s*)] is computed and the root-mean-square error *RMSn* between [*He*(*s*)] and [*Hn*(*s*)] in the subband is again calculated and stored.

7) If *RMSn* < *RMSe*, then old poles and residues are replaced with new poles and residues. If *RMSn* > *RMSe*, then old poles and residues are retained.
8) Steps 1 to 7 are repeated until the error is minimized.

The location and size of each subband is recalculated iteratively in the pole replacement method. The error criterion determines the maximum error value *Emo*, the location *Lmo*, and the width of the subband being recalculated. BEMP uses the root-mean-square error as the error criterion while the subband width around the location *Lmo* is determined based on the position of poles being replaced. Assuming the number of poles being replaced is k, the algorithm searches for (k +2) poles around the location *Lmo*. The (k +2) poles are then reordered in increasing frequency such that pole $p(1)$ corresponds to the lowest frequency and pole $p(n)$ corresponds to the highest frequency. The left and right boundaries of the subband are then determined as the mid-frequency points between poles $p(1)$, $p(2)$ and $p(n\text{-}1)$, $p(n)$, respectively. If the width and position of the subband remains the same as before, then the subband is suitably dilated to minimize error. If the location *Lmo* is not changed after dilating the subband, then the location *Lmo* is stored and ignored in subsequent iterations. With the use of band division, subband reordering and subband dilation methods initially, the required number of iterations for the pole replacement method locally is minimum. Even though the pole replacement method was originally intended for increasing the accuracy of poles and residues of passive macro-models, it can be used to compensate for negative eigenvalues by inserting additional poles and residues, as discussed earlier. After the poles and residues in the entire frequency band or subbands are calculated, the residue matrices $[\delta]$ and $[\eta]$ can be calculated. It is important to note that broadband macro-models can be constructed using band division, subband reordering, subband dilation, and pole replacement methods along with frequency scaling, without having an ill-conditioned matrix problem. In addition, since the number of required poles is reduced and the orders *NS* and *DS* become small within a subband, the size of the matrix $[A]^T$ $[A]$ becomes small and the required computational memory and CPU time can be reduced.

3.1.5 Network Synthesis

Using the pole-residue representation of the rational function representing the admittance parameters, electrical networks consisting of resistors, inductors, capacitors, and controlled sources can be constructed. In (5) and (11), RLC networks can be used to represent complex conjugate poles and residues of the band pass filter, RL networks can be used to represent real poles and residues of the low pass filter, and RC networks can be used to represent real poles and residues of the high pass filter. A resistor and a capacitor can be used to represent the residues of the all pass filter. The electrical network configurations are shown in Fig. 3.5 and the values of the components are shown in Fig. 3.6. It is important to note that the frequency scaling in (19) and the local ground for SPICE sub-circuits have been used in the circuit implementation.

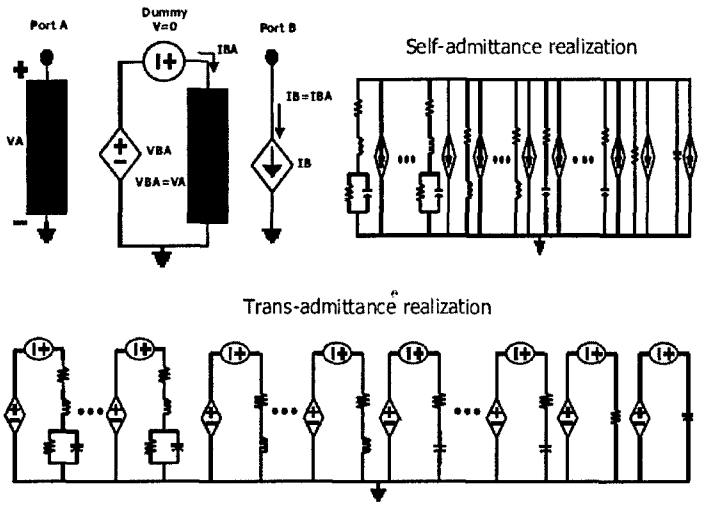


Fig. 3.5: Electrical network configurations for macromodels using admittance parameters

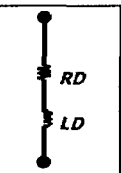
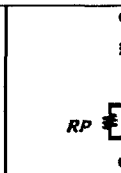
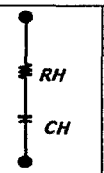
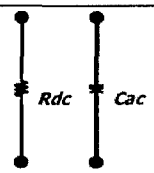
Low-pass filter	Band-pass filter	High-pass filter	All-pass filter
$Y_m(s)=\frac{\gamma_m}{s-p_m}$	$Y_n(s)=\frac{2\alpha_n(s-p_{nr})-2\beta_n p_{ni}}{(s-p_m)^2+p_{ni}^2}$	$Y_h(s)=\frac{s\psi_h}{s-p_h}$	$Y(s)=\delta \quad Y(s)=\eta s$
			
$RD_m=\frac{-p_m}{\gamma_m}$ $LD_m=\frac{1}{\omega_s\gamma_m}$	$RS=\frac{-\alpha_n p_{nr}+\beta_n p_{ni}}{2\alpha_n^2}$ $LS=\frac{1}{2\omega_s\alpha_n}$ $RP=\frac{p_{ni}^2(\alpha_n^2+\beta_n^2)}{2\alpha_n^2(-\alpha_n p_{nr}-\beta_n p_{ni})}$ $CP=\frac{2\alpha_n^2}{\omega_s p_{ni}^2(\alpha_n^2+\beta_n^2)}$	$RH=\frac{1}{\psi_h}$ $CH=-\frac{\psi_h}{p_h\omega_s}$	$Rdc=\frac{1}{\delta}$ $Cac=\frac{\eta}{\omega_s}$

Fig. 3.6: The values of elements (electrical representations of macro-models)

3.2 Broadband Efficient Macro-modeling Program (BEMP)

3.2.1 BEMP version 3.0

The Broadband Efficient Macro-modeling Program (BEMP) incorporates the entire modeling process described in the previous section. BEMP has been coded using the C++ programming language and is executable on the Windows platform. The program inputs S, Y, or Z parameter representations of multi-port networks and generates SPICE sub-

circuit files that can be directly incorporated into a SPICE simulation. The *Touchstone* format is used to input data into BEMP. This format has been chosen so that both measurement and simulation data can be easily and efficiently modeled using BEMP. Most field solvers and all vector network analyzers have the option of storing data in the *Touchstone* format. Details on the format are available in the BEMP manual.

The input data in *Touchstone* format when read by BEMP is first converted to a Y-parameter representation through simple matrix operations. All macro-modeling techniques described in the previous section are then performed on the Y-parameters. Fig. 3.7 illustrates how BEMP goes about generating a SPICE net list form an input *Touchstone* file containing multi-port frequency response data for a passive network. The Fig. begins with explaining the algorithm after the input data has been converted to the Y-parameter format.

INITIALIZATION:
load original admittance parameters ***Ho*** and the frequency ***fo***
load subbands ***FSS*** (Band Division)
load estimated orders ***NS*** *and* ***DS***
initialize poles and residues
LOCAL LOOP:
for n=1 **to** *the number of subbands*
  set sub-computational domain ***SCD*** = ***FSS***(n) (Subband Reordering)
  apply the band-rejection selector on poles and resides
  calculate the frequency response ***Hm*** using poles and residues over the ***SCD***
  calculate ***He*** = ***Ho*** - ***Hm*** and **save** ***Hg*** = ***He*** and max***Hg*** over the ***SCD***
  initialize good-sub-poles and good-sub-residues
  for num, den = a range of estimated orders ***NS*** and ***DS*** ([8])
    initialize sub-poles and sub-residues
    do frequency scaling on ***fo*** (Frequency Scaling)
    calculate the denominator coefficients using the eigenvalue method ([8])
    construct stable sub-poles from the denominator coefficients
    undo frequency scaling on stable sub-poles
    calculate sub-residue matrices of sub-poles
    calculate the residues *[δ]* and *[η]*
    apply multiport passivity formulae
    if there is a violation, **compensate** negative eigenvalues, **end**
    calculate the frequency response ***Hm*** using sub-poles and sub-residues
    calculate ***Hs*** = ***He*** - ***Hm*** over the ***SCD***
    if max***Hg*** > max***Hs*** based on error criterion
      save max***Hg*** = max***Hs***
      save good-sub-poles = sub-poles, good-sub-residues = sub-residues
    end
    apply appropriate selectors on good-sub-poles and good-sub-residues
    add good-sub-poles and good-sub-residues **to** poles and residues
  end
  apply subband dilation and pole replacement methods over the ***SCD***
end
GLOBAL LOOP:
for n=1 **to** *the number of global optimization*
  apply the pole replacement method over the entire frequency band
  update the residues *[δ]* and *[η]*
end
generate SPICE net lists

Fig. 3.7 A flow chart of BEMP

The application of the modeling process in BEMP as described previously is graphically illustrated in Fig. 3.8. The exact poles of the network are shown at the top of the Fig.. It has been assumed that all the poles are complex conjugate poles. In Fig. 3.8, the good poles are the exact poles of the network. All poles except the good poles are regarded as spurious poles. Using the band division method, the entire frequency response has been divided into 7 non-uniform subbands such that each subband has between one to four resonant peaks. The bands are numbered from 1 - 7 horizontally and named A - G vertically. This has been done intentionally to differentiate band division from subband reordering. The steps illustrated below assume that after the extraction of the poles from each subband, the corresponding residues are extracted. The frequency response of the subband macromodel is then subtracted from the overall response prior to the macromodel construction for the next subband. The various steps are described below:

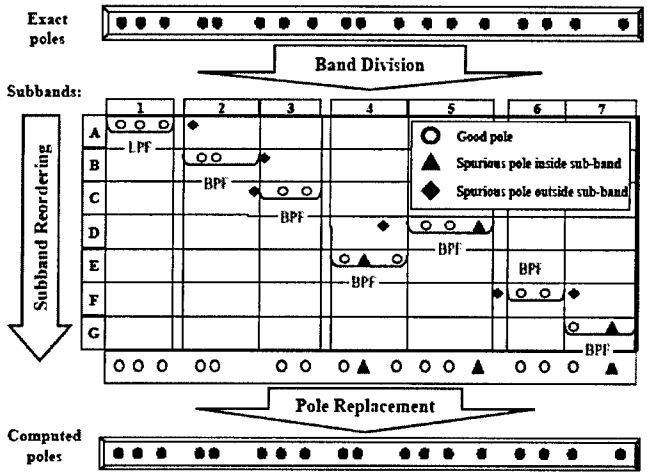


Fig. 3.8 Illustration of band division and sub-band reordering

1) During the construction of sub-macro-models from subband A, three good poles and a spurious pole located outside subband A are extracted using the eigenvalue method discussed earlier. After applying the low pass selector with bandwidth equal to subband 1, three good poles are extracted.
2) After subtracting the frequency response of the above macromodel from the overall response, a spurious pole located outside subband B and two good poles are extracted from subband B. After applying the band pass selector on subband B, the spurious pole is removed and two good poles are retrieved. Note that there is an exact pole near the boundary between subbands B and C, which is not included.
3) During the calculation in subband C, a spurious pole located outside subband C and two good poles are extracted. After applying the band pass selector, two good poles are retained. Up to this point, seven good poles have been extracted and an exact pole near the boundary of subbands B and C has been missed.

4) The next computational domain moves to subband D corresponding to subband 5 instead of subband 4 (subband reordering). This is because the magnitude of the frequency response is larger in subband 5 than in subband 4. Sub-macro-models constructed from subband D result in two spurious poles and two good poles. After applying the band pass selector to remove poles located outside subband D, a spurious pole and two good poles are extracted.
5) From subband E corresponding to subband 4, a spurious pole and two good poles are extracted. After applying the band pass selector, a spurious pole and two good poles are retained.
6) From subband F corresponding to subband 6, two spurious poles located outside subband F and two good poles are extracted. After applying the band pass selector, two good poles are retained.
7) From subband G, a spurious pole and a good pole are extracted after applying the high pass selector. After collecting sub-macro-models from each subband, 14 good poles and 3 spurious poles are found using the band division and subband reordering methods. The pole replacement method is now applied to correct the good poles and eliminate the spurious poles.

3.2.2 Additional Features: Enforcement of causality

The macro-models obtained using BEMP are guaranteed to be causal and stable over all frequencies. However, for timing analysis in high-speed digital design, especially involving long delay interconnects; causality of the macro-models is a major issue. Causality deals with the precise timing of signal propagation through passive structures like interconnects and power distribution networks. If unaccounted for it can lead to considerable error in the signal integrity analysis of networks. Hence the macro-models obtained using BEMP need to be made causal. This work on enforcing causality is still in progress and is documented in the following sections. The method described in this section will be implemented in BEMP v4.0.

3.2.2.1 Delay extraction from frequency domain data

With increasing clock frequencies in digital and mixed-signal systems, the size of the passive structures is becoming comparable to the signal wavelength at the operating frequency, leading to distributed effects like delay playing an important role in time domain analysis. These distributed effects imply that there are many causality conditions that need to be satisfied by a passive macro-model, to generate the correct signal response in the time domain. These causality conditions depend on the delay in the passive network, which is the time taken by the signal to travel from one port to another. Hence determining this delay from the frequency domain response is an important step towards causal macro-modeling.

Using the minimum-phase property of the response functions of passive networks, a transfer impedance response $Z12(j\omega)$ can be separated into a minimum phase function and an all-pass function as follows [4]:

$$|Z12'(j\omega)| = |Z12(j\omega)| \tag{20}$$

$$\arg[Z12'(j\omega)] = -\frac{1}{2\pi}\mathrm{P}\int_{-\pi}^{\pi}\log|Z12(j\theta)|\cot\left(\frac{\omega-\theta}{2}\right)d\theta \tag{21}$$

$$e^{-j\omega Td} = \frac{Z12(j\omega)}{Z12'(j\omega)} = Z_{AP}(j\omega) \tag{22}$$

$$Td = -\frac{\arg(Z12_{AP}(j\omega))}{\omega} \tag{23}$$

where *Z12'(jω)* is the minimum phase portion of *Z12(jω)* and $Z12_{AP}(j\omega)$ is the all-pass portion of *Z12(jω)*. The negative gradient of the phase response of $Z12_{AP}(j\omega)$ gives the delay embedded in the response *Z12(jω)*. This methodology can be used to extract delay from the S, Y and Z parameter representations of passive networks.

Mixed-mode passive structures like differential transmission lines are characterized by two types of delays: differential mode (odd mode) delay and common mode (even mode) delay. The technique discussed above can be easily extended to extract the even and odd mode delays in mixed-mode systems. The process involves transforming the given system parameters into mixed-mode parameters and applying the above technique on the new set of mixed mode parameters.

3.2.2.2 Causality Compensation and Results

The delay extracted from the frequency response can be used for enforcing causality on the system response. To compensate a frequency response for causality, the following steps are carried out

1. The frequency response is converted into the time domain.
2. The delay in the response is extracted as explained in the previous section.
3. The extracted delay is used to force the time domain response to zero over the delay period.
4. The signal energy in the response is adjusted to account for the loss due to step 3. This ensures correct steady state response.

This has been demonstrated through the simulation of a long transmission system. A microstrip transmission line 50cm in length and with a characteristic impedance of 50Ω was constructed in Agilent's Advanced Design System (ADS). The line was terminated with a 50Ω load at the far end and was excited at the near end with a 250mV/100ps step source having 50Ω input impedance. Fig. 3.9 shows the response obtained at the far end with and without causality compensation enforced on the transmission line macro-model. The response without causality compensation shows a violation of causality.

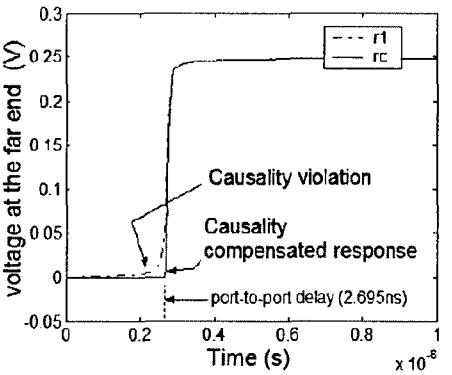


Fig. 3.9 Causality compensation

Next the transmission line was excited by a random bit-pattern and an eye-diagram was obtained for the far end response. Fig.s 3.10(a) and (b) show the eye-diagrams obtained with and without causality compensated frequency domain data.

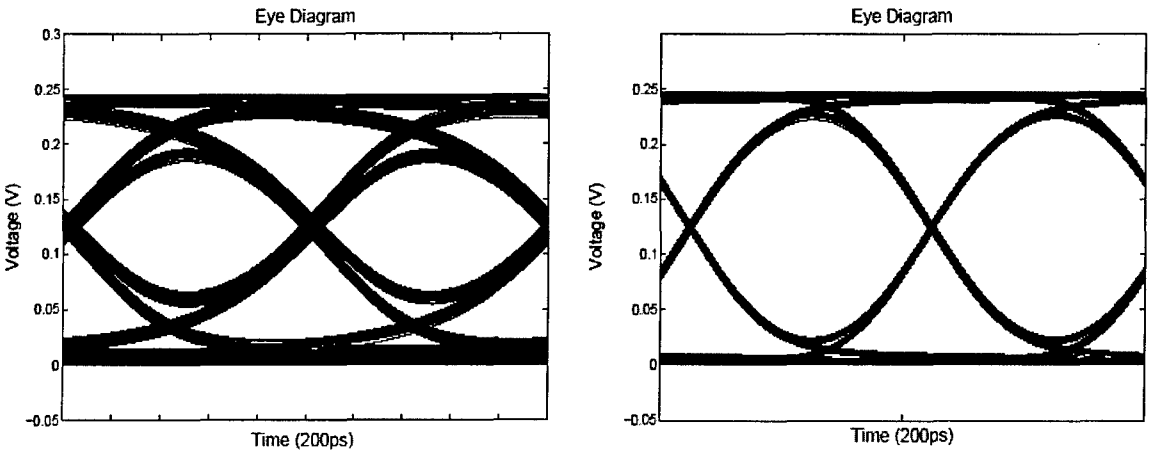


Fig. 3.10 Eye-diagrams using uncompensated (a) and causality compensated (b) macro-models

3.2.3 Additional Features: Nonlinear Macro-modeling Of I/O Drivers And Receivers

Signal integrity (SI) and timing analysis of large digital systems in today's world is becoming more and more complex both in terms of CPU memory required and time consumed for simulation. One method to reduce this complexity is to use macro-models of digital driver and receiver circuits constituting these large digital systems. Accurate modeling of the digital drivers to capture their non-linearity is a big challenge. Black-box modeling techniques are helpful in modeling digital driver circuits since they are independent of the knowledge of the internal logic of the driver [4]. Fig. 3.11 shows a 2-port black-box circuit model of a digital buffer with information about its input and output voltage-current characteristics and no information about its internal circuitry.

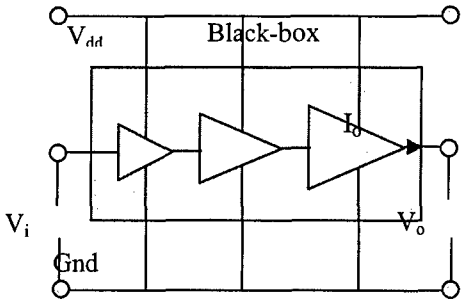


Fig. 3.11. Black-Box representation of a driver circuit

In this report, a modeling approach using spline functions with finite time difference approximation is used for modeling moderately non-linear digital driver circuits. To estimate power supply noise and cross-talk accurately, the method has been extended to multiple ports. However, for highly non-linear circuits, this method has problems with accuracy. To alleviate this problem, a method based on Recurrent Neural Networks (RNN) has been developed. Also in this report, a methodology based on RNN and static characteristics with finite time delay element is used to macro-model receiver circuits. The accuracy and speed-up of the macro-models has been compared against the transistor level circuit models on some test cases. The non-linear macro-models is an extension of the linear macro-models described in the previous section and will be integrated in BEMP v4.0.

3.2.3.1 Spline Function With Finite Time Difference Approximation Modeling Technique

The output current of any driver can be expressed as a function of the output voltage using static characteristics. Let sub-models f_{1s} and f_{2s} represent the static characteristic relation between the driver output current and the driver output voltage for a driver when the driver input is set HIGH and LOW respectively, as shown in equation (24)

$$f_{sn}(k) = A_{nm} v_O^{\,m}(k) + A_{n(m-1)} v_O^{\,m-1}(k) + ... + A_{n0}, \; n=1,2; \; m \geq 1 \tag{24}$$

where, As are constants, v_o is the driver output voltage and the value of m is usually less than 5. The static sub-models f_{1s} and f_{2s} can be estimated by conducting a *dc* sweep at the output of the driver for both driver inputs HIGH and LOW respectively. Static characteristic models do not however capture the nonlinearity associated with the dynamic response of the driver. Since static characteristic relations fail in capturing the dynamic characteristics, static sub-models in (24) can be modified to include the previous time instances of the driver output current so that dynamic behavior of the driver can be captured.

When the driver input is set HIGH, the current at the output, i_{oh}, can be expressed as sub-model f_{1s} as shown in equation (25)

$$f_{1s}(k) = i_{oh}(k) = A_{1m} v_O(k)^m + A_{1(m-1)} v_O(k)^{m-1} + ... + A_{10} \tag{25}$$

Sub-model f_{1s} at time instance '*k-1*' can be expressed as equation (26)

$$f_{1s}(k-1) = i_{oh}(k-1) = A_{1m}v_O(k-1)^m + A_{1(m-1)}v_O(k-1)^{m-1} + \ldots\ldots + A_{10} \quad (26)$$

Incremental change in the driver output current Δi_{oh} is the difference between the present instance *(k)* and previous time instance *(k-1)* values of sub-model f_{1s} as shown in equations (27) and (28)

$$f_{1s}(k) - f_{1s}(k-1) = i_{oh}(k) - i_{oh}(k-1) = \Delta i_{oh} \quad (27)$$

$$\text{(Or) } f_{1s}(t) - f_{1s}(t-\Delta t) = \Delta i_{oh} \quad (28)$$

Once Δi_{oh} is calculated, first derivative of driver output current i_{oh}' can be approximated using equation (29) as

$$\frac{f_{1s}(t) - f_{1s}(t-\Delta t)}{\Delta t} = \frac{\Delta i_{oh}}{\Delta t} = i'_{oh} \quad (29)$$

where, Δt is the sampling time. The effect of dynamic behavior is captured in i_{oh}', similarly the effect of dynamic behavior for f_2 is captured by $\Delta i_{ol}'$. Therefore, static sub-models f_{1s} and f_{2s} can be modified as

$$\begin{aligned} f_{1n}(k) &= f_{1s}(k) + p * i'_{oh} \\ f_{2n}(k) &= f_{2s}(k) + pp * i'_{ol} \end{aligned} \quad (30)$$

where p and pp are constants whose magnitude can be estimated by calculating the error between f_{1s}/f_{2s} and transistor level driver output current values for inputs HIGH/LOW respectively. It is important to note that there is no limitation on the number of previous output current time instances in the static sub-models f_{1s}/f_2 [5].

Once f_{1n} and f_{2n} are estimated for input HIGH and LOW respectively, the relation between driver output current and voltage can be expressed as shown in equation (31)

$$i_O(k) = w_1(k)f_1(k) + w_2(k)f_2(k) \quad (31)$$

where, w_1 and w_2 are the weighting functions that help f_1 and f_2 transition from one state to another. Since, two unknowns exist, weighting functions w_1 and w_2 can be found by linear inversion of (31) for two different loads.

3.2.3.2 Extension To Multiple Ports

To incorporate the effect of the power supply node v_{dd}, a new relation should be drawn between driver power supply current (i_{dd}) and driver power supply voltage [6]. However, the driver power supply current is not only a function of driver power supply voltage but also a function of driver output voltage (v_o), as shown in equation (32)

$$i_{dd}(k) = w_{1dd}(k)f_{1dd}(v_o(k), v_{dd}(k)) + w_{2dd}(k)f_{2dd}(v_o(k), v_{dd}(k)) + w_{3dd} \quad (32)$$

where, f_{1dd} and f_{2dd} are the power supply sub-models that relate the power supply current to power supply voltage for input HIGH and LOW respectively. In (32), weighting functions w_{1dd}, w_{2dd}, and w_{3dd} help sub-models f_{1dd} and f_{2dd} in transitioning from one state to another. Sub-models f_{1dd} and f_{2dd} can be expressed as a combination of static sub-models (f_{1sd} and f_{2sd}) and dynamic sub-models (f_{1dy} and f_{2dy}), as shown in equation (33)

$$f_{ndd}(v_o(k)) = f_{nsd}(v_o(k), v_{dd}(k)) + f_{ndy}(v_o(k), v_{dd}(k)) \text{ ; where, } n = 1,2 \quad (33)$$

The static sub-models f_{1sd} and f_{2sd} are calculated through a double *dc* sweep at driver output and at driver power supply. This relationship is shown in equation (34) as:

$$f_{nsd}(v_o(k)) = A_{nr} v_o{}^{r}(k) v_{dd}{}^{s}(k) + A_{n(r-1)} v_o{}^{(r-1)} v_{dd}{}^{(s-1)}(k) + \ldots + A_{no} \quad (34)$$

where, n = 1,2;r >=1;s >=1;

The dynamic sub-models f_{1dy} and f_{2dy} are expressed in equation (35) as:

$$f_{ndy}(v_o(k)) = p_{dn}\Delta i_{on}' + q_{dn}\Delta i_{ddn}' + pp_{dn}\Delta i_{on}'' + qq_{dn}\Delta i_{ddn}'' \ldots \quad ; n = 1,2 \quad (35)$$

In equation (35), dynamic characteristic sub-model (f_{ddn}) constants pd_n, qd_n, ppd_n and qqd_n are estimated by connecting PWL voltage sources both at driver output and at driver power supply and measuring the error between static power supply current and transistor level driver power supply current. In equations (34) and (35), A_n, p_{dn}, q_{dn}, pp_{dn} and qq_{dn} are all constants and depend on the driver being modeled. Terminating the driver with 2 different loads result in two equations of (31) and since, three unknown weighting functions exist, one method for solving the problem is by assuming $w_{1d} = (1 - w_{2d})$. Thus, weighting functions w_{1d}, w_{2d} and w_{3d} can be calculated once f_{1d} and f_{2d} are estimated for 2 different loads. Similarly, driver output current is not only a function of driver output voltage but also a function of driver supply voltage. The procedure for estimating driver output current is similar to estimating driver power supply current.

Test case 1: A test case is generated where an IBM driver (HSTL_B) was connected to a plane pair which was modeled using the cavity resonator method. The plane pair was 10 cm × 6 cm in length and width. It had 5 ports on each plane, V_{dd} and *Gnd* as shown in Fig. 3.12. Ten drivers were connected at port 1 and all the drivers were connected to 25 ohm transmission lines which in turn were terminated using 1 pF capacitance. All transmission lines were terminated at port 3. The power supply (v_{dd}) was at port 4. Two additional ports were used for probing. All 10 drivers were identical. The resulting simultaneous switching noise (SSN) was calculated using both actual transistor level driver model and spline function with finite time difference approximation model. Fig. 3.13(a) shows the near end and far end voltage waveforms on transmission line 1. It can be clearly seen that the modeled and the transistor level model waveforms match accurately. A third order spline function with two previous time instances was used to model sub-models f_1 and f_{1dd}. Fig. 3.13(b) shows the SSN at ports 1, 3 and 5 respectively both for modeled and actual simulated cases and again it can be clearly seen that the noise is estimated accurately using the spline function with finite time difference approximation method. Spline function model is 8 times faster than the IBM transistor level driver model.

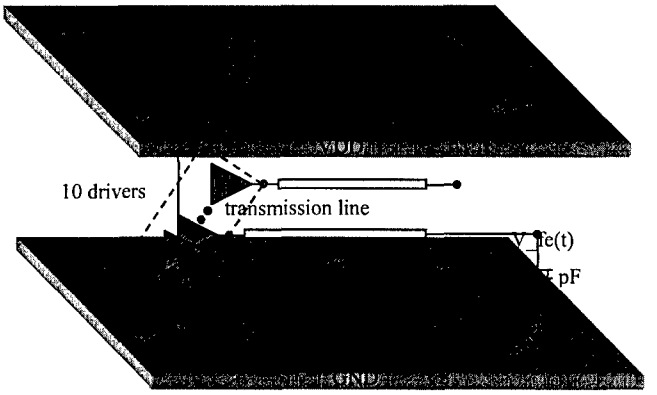


Fig. 3.12. Plane pair model generated using cavity resonator method. Both planes have 5 ports each.

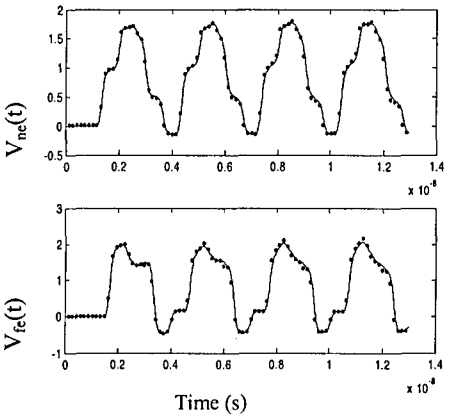


Fig. 3.13(a). Near end (V_ne(t)) and far end (V_fe(t)) voltage waveforms on transmission line # 1 for IBM transistor level driver model (straight line) and spline function with finite time difference approximation model (dotted line).

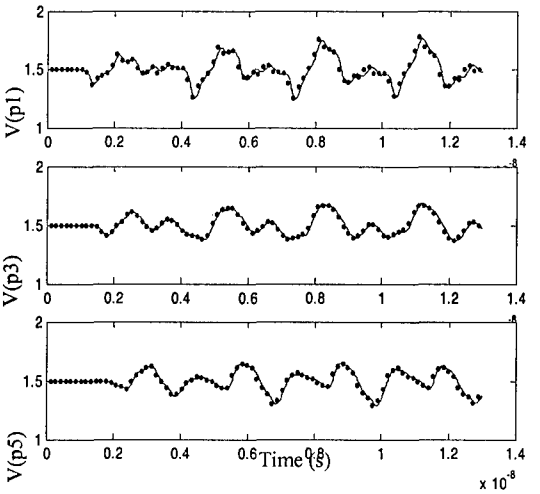


Fig. 3.13(b) Simultaneous Switching Noise (SSN) at ports p1, p2 and p5 when 10 identical drivers are switching together. SSN from actual transistor level driver model (straight line) and spline function with finite time difference approximation model (dotted line).

3.2.3.3 Recurrent Neural Network Modeling Technique

Spline function with finite time difference approximation has some limitations. When the transistor level driver circuit models are highly complex, the proposed method cannot capture the high non-linearity present in the driver. One solution for modeling these highly non-linear driver circuits is by using Recurrent Neural Networks (RNN). Sub-functions f_1 and f_2 are now expressed using RNN functions as shown in equations (36) and (37)

$$f_n = \sum_{k=1}^{M} b_{kj} g\left(\sum_{j=1}^{N} a_{ji} x_i + a_{oj}\right) + b_{ok} \; ; n = 1,2 \tag{36}$$

$$g(u) = (e^{x} + e^{-x})/(e^{x} - e^{-x}) \tag{37}$$

where, b and a are weights associated with the neural network, N represents number of hidden neurons, M represents number of outputs and x is the regressor vector that takes into account both the past and present samples of driver output current, output voltage, power supply current and power supply voltage as shown in (38).

$$x(k) = \begin{Bmatrix} i_o(k-1), i_o(k-2), \ldots, i_o(k-r) \\ v_o(k), v_o(k-1), \ldots, v_o(k-r) \\ i_{dd}(k-1), i_{dd}(k-2), \ldots, i_{dd}(k-r) \\ v_{dd}(k), v_{dd}(k-1), \ldots, v_{dd}(k-r) \end{Bmatrix} \tag{38}$$

A RNN is a special type of neural network having the capability of learning and then representing dynamic system behavior. RNN has been used in areas such as control, signal processing and system identification. A training method based on back propagation through time (BPTT) is used for training the RNN.

Test Case 2: A test case with four IBM drivers ('DDR2') connected to a small plane pair modeled using cavity resonator method was generated. IBM 'DDR2' driver has a power supply voltage of 2.5 volts and it is usually driven at 250 MHz with a rise time of 1.25 ns. The pane pair has a dimension of 6 cm X 4 cm with 4 ports on V_{dd} plane and 4 ports on *Gnd* plane. All the drivers were identical, driving a 50 ohm transmission line and were connected at port 1. The power supply node was at port 3. All the transmission lines were terminated at port 2 using a 2 pF capacitance and port 4 was used for probing. Fig. 3.14(a) shows the plane pair used to model the power supply noise when 4 drivers are simultaneously switching. The IBM driver was modeled using RNN. The regressor vector x consists of present samples of power supply voltage and driver output voltage and past samples of driver output current, power supply current, power supply voltage and output voltage. The RNN model for sub-functions $f_{1,2}$ and sub-functions $f_{1d,2d}$ require one hidden layer with 2 hidden neurons. BPTT training algorithm was used to estimate the weights of the RNN model. A third order spline function macro-model with two time instances back was generated. Power supply noise at ports 1, 2 and 4 was measured for RNN model, spline function with finite time difference approximation model and transistor level driver model. It can be seen from Fig. 3.14(b) that RNN model captures SSN accurately when all 4 drivers are switching simultaneously, whereas, spline function

with finite time difference model fails to accurately model SSN. RNN model is 6-7 times faster and spline model is 13-15 times faster than transistor level driver model, when the time domain simulations are carried out for 3 cycles. The computational time speed up between the RNN model and actual transistor level model increases with the increase in complexity of the circuit.

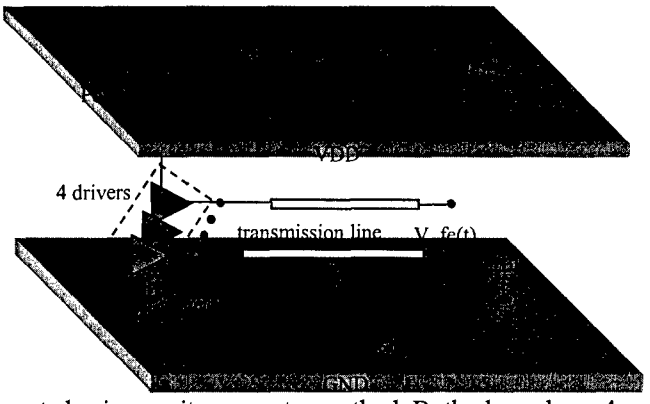


Fig. 3.14(a) Plane pair model generated using cavity resonator method. Both planes have 4 ports each.

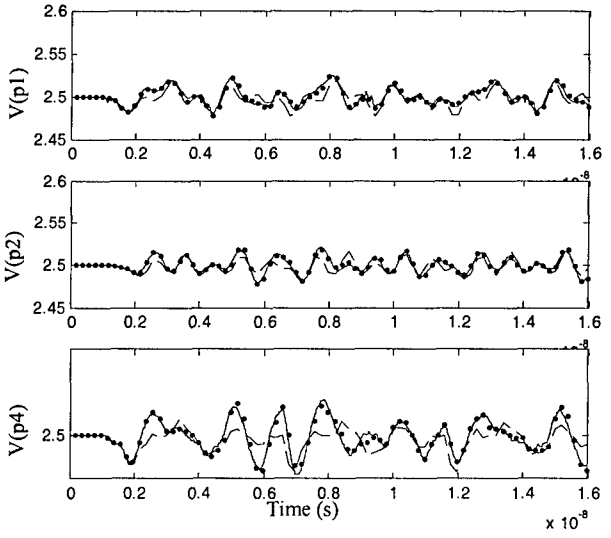


Fig. 3.14(b) Simultaneous Switching Noise (SSN) at ports p1, p2 and p4 when 10 identical drivers are switching together. SSN from actual transistor level driver model (straight line) RNN model (dotted line) and spline function with finite time difference approximation model (dashed line).

3.2.3.4 Receiver Modeling Methodology

In order to model the input characteristics of the receiver accurately, the input current, i_{in}, should be accurately modeled. The input current can be expressed as equation (39)

$$i_{in}(k) = f_{in}(x(k)) \tag{39}$$

$$x(k) = \begin{Bmatrix} i_{in}(k-1), i_{in}(k-2), \ldots, i_{in}(k-p), \\ v_{in}(k), v_{in}(k-1), \ldots, v_{in}(k-p) \end{Bmatrix} \tag{40}$$

where, $x(k)$ takes into account all the previous time instances of receiver input voltage, v_{in}, and input current, i_{in}. Function f_{in} is a non-linear RNN function as shown in equation (41)

$$f_{in}(x) = \sum_{k=1}^{M} b_{kj} g(\sum_{j=1}^{N} a_{ji} x_i + a_{oj}) + b_{ok} \tag{41}$$

$$g(x) = (e^{x} - e^{-x})/(e^{x} + e^{-x}) \tag{42}$$

where, a and b are weights associated with the recurrent neural network, N represents number of hidden neurons, M represents number of outputs. The weights are estimated using Back Propagation Through Time (BPTT) algorithm. To estimate f_{in} accurately a good training data is required that should excite both the linear and the non-linear regions of the receiver input current [7].

3.2.3.5 Receiver Output Characteristics

The output voltage of the receiver is modeled using a combination of static receiver characteristics ($v_{in} - v_{out}$) and a delay element to capture the input-output delay of the receiver. The input-output delay of a receiver is dependent on the slew rate of the input voltage and independent of the frequency of excitation for normal range of operations. The delay for the particular slew rate can be easily estimated and the relation between input and output voltages can be captured using the static relation. Thus, the combination of the two would result in accurate modeling of the receiver output voltage.

Test Case:

Test Case 3: A realistic test case was setup where an IBM DDR2 driver was connected to a 75 ohm ideal transmission line with a 0.2ns time delay. The transmission line was in turn connected to a DDR2 receiver. A RNN model was used to model the DDR2 driver and DDR2 receiver was modeled using the proposed technique. The driver was excited with a 2.5V pulse with 0.25ns rise time. The voltages at the near end of the transmission line, the input of the receiver and the output of the receiver were plotted as shown in Fig. 3.15. It can be clearly seen that the modeled results match well with the IBM DDR2 transistor level receiver model. The transistor level DDR2 setup takes 290 seconds for computation and the macro-model takes less than 4 seconds for simulation. The macro-model is very accurate and gives a timing error of less than 20 ps

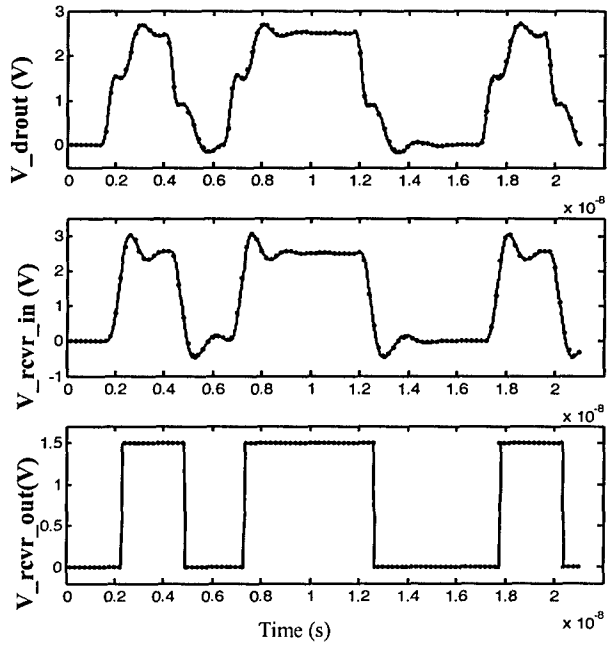


Fig. 3.15 Driver output for IBM DDR2 (straight line) and for macro-model (dotted line); Receiver input for IBM DDR2 (straight line) and for macro-model (dotted line); receiver output for IBM DDR2 (straight line) and for macro-model (dotted line).

3.2.3.6 Non-linear macro-modeling comparison

Test Case Description	Simulation Time Consumed		Speed-up
	Transistor Level Circuit Model	Macro-Modeling Technique	
IBM AGP Driver connected to a 50 ohm transmission line	771 s	Spline function model (7 s)	110X
16 IBM HSTL_A driver connected to a plane pair for SSN estimation	3096 s	Spline function macro-model (150 s)	20X
10 IBM HSTL_A driver connected to a plane pair for SSN estimation	550 s	Spline function macro-model (70 s)	8X
4 IBM DDR2 receiver connected to a plane pair for SSN estimation	560 s	RNN model (80 s)	7X
IBM DDR2 receiver circuit connected to a 50 ohm	96 s	Statis model with finite time delay element	25X

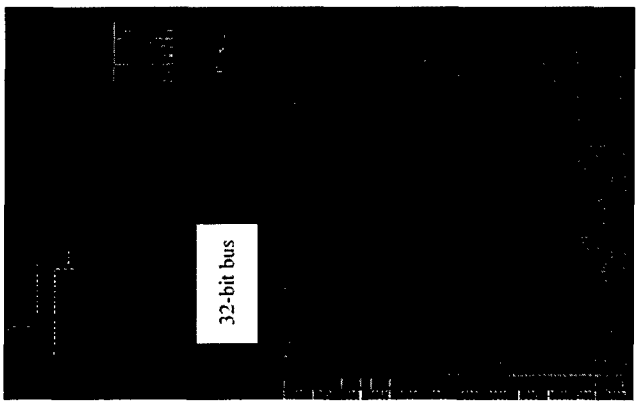
transmission line		(4 s)	
IBM DDR2 driver and receiver circuits connected to a 50 ohm transmission line	290 s	RNN macro-model for driver and static model with finite time delay element (4 s)	70X

3.3 Modeling Results and Benchmarking

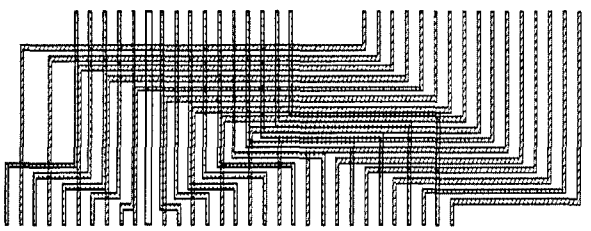
a) A 32 bit bus:

BEMP was used in macro-modeling of a 32-bit wide interconnect bus on a Delta-Sigma modulator chip designed by HRL. Fig. 3.9(a) shows the interconnects (32-bit bus) on a Delta-Sigma modulator chip deigned by HRL. The 32-bit bus shown in Fig. 3.9(b) was modeled using an EM simulator that is proprietary to HRL. Therefore, only the frequency response of the bus was provided. Using this information, a macromodel of the 32-bit bus was developed. The comparison of the magnitude in logarithm scale between the given frequency response of the bus and the frequency response of the macromodel using BEMP version 3.0 is shown in Fig. 3.10. This result in Fig. 3.10 is for Y(1,19), which was screen-captured when BEMP version 3.0 was run on Windows. The deviation (RMS error) in Fig. 3.10 shows the accuracy of the macromodel. For this test case, it is important to note that only diagonal elements were used for the approximation of 32-port admittance parameters. Hence, as compared to an original computation time of $C = O(K \text{ x } N^6) = O(K \text{ x } 1{:}0737 \text{ x } 10^9)$, where $N = 32$ and K is a constant, a computation time of $C = O(K \text{ x } N^3) = O(K \text{ x } 32768)$ was achieved, which resulted in a speed up of 32767.

The 32-port macromodel was connected to a 3.3V dc source having 100ps rise and 200ps fall times at port 1, as shown in Fig. 3.11. Fig. 3.12 show the time-domain waveform when all the ports were terminated with 30- resistors, which was used to simulate the crosstalk between ports 1 and 9.



(a)



(b)

Fig. 3.9 (a) Layout of Delta-Sigma modulator from HRL and (b) layout of 32-bit bus

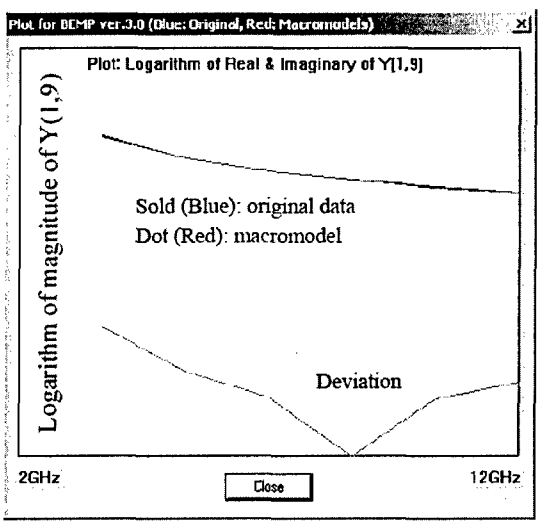


Fig. 3.10: Comparison of admittance parameter Y(1,9)

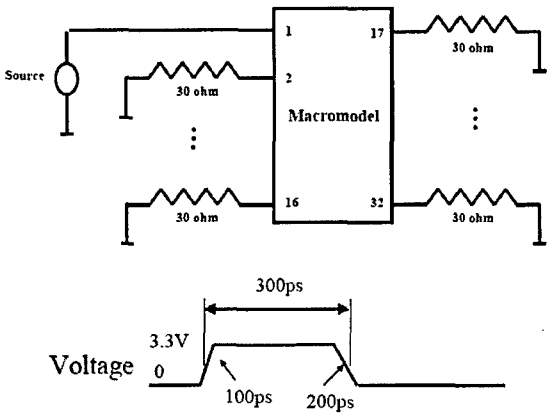


Fig. 3.11 Circuit being simulated

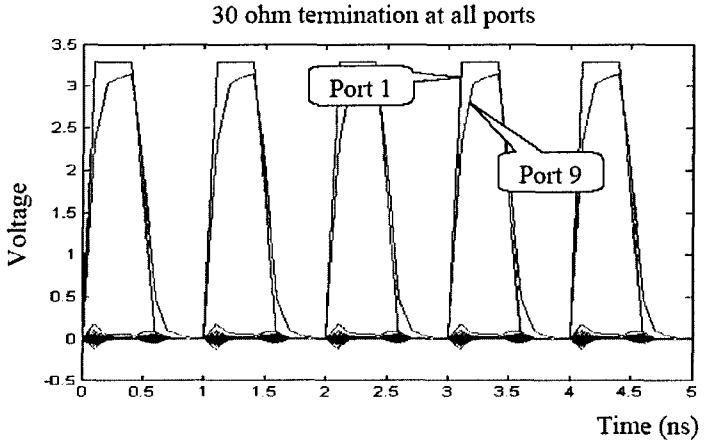
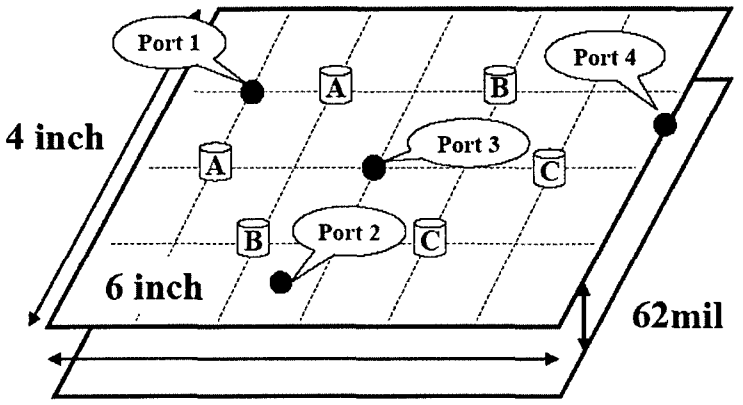


Fig. 3.12: Crosstalk between ports 1 and 9

b) A 4-port power plane pair with de-coupling capacitors:

A 4.0in x 6.0in power plane pair with a dielectric thickness of 62mil is shown in Fig. 3.13. Six decoupling capacitors represented as RLC circuit elements were connected to the power plane. A total of 1000 uniformly spaced samples were generated from 1.0 MHz to 4.0 GHz using the cavity resonator model. Although the frequency response of the power plane pair with decoupling capacitors is highly irregular, the entire frequency domain for admittance parameters was divided into 20 uniform subbands having 50 samples per subband without overlapping subbands. The number of local and global iterations for the pole replacement method was 1 and 200, respectively. Using the orders *NS* of 4 and *DS* of 5 within each subband, the number of complex conjugate poles and real poles extracted was 78 and 1, respectively. The comparison between original data and the frequency response of the constructed macromodel for admittance parameters over a bandwidth of 4.0 GHz is shown in Fig.s 3.14 and 3.15.



Dielectric Properties: Er=5.05 Loss tangent=0.018 Conductivity=5.7e+7
Decoupling Capacitors:

Fig. 3.13: Power plane pair with decoupling capacitors

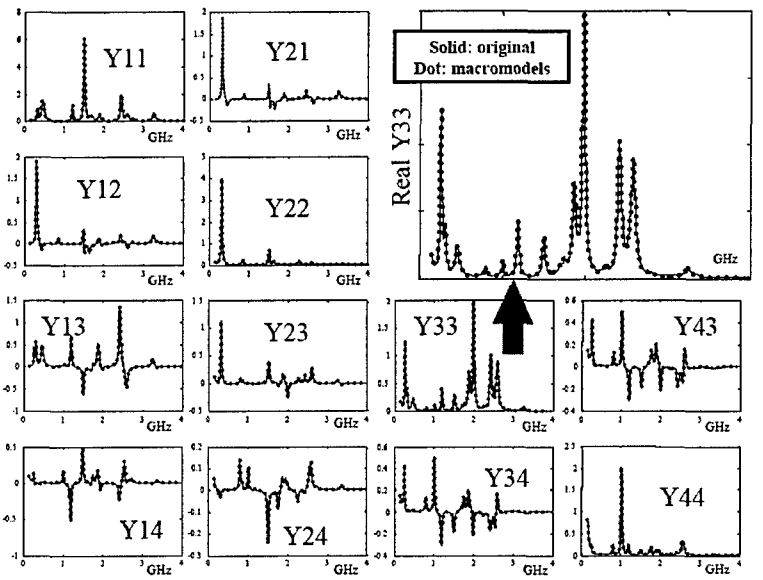


Fig. 3.14: Comparison of real part of admittance parameters for the power plane pair

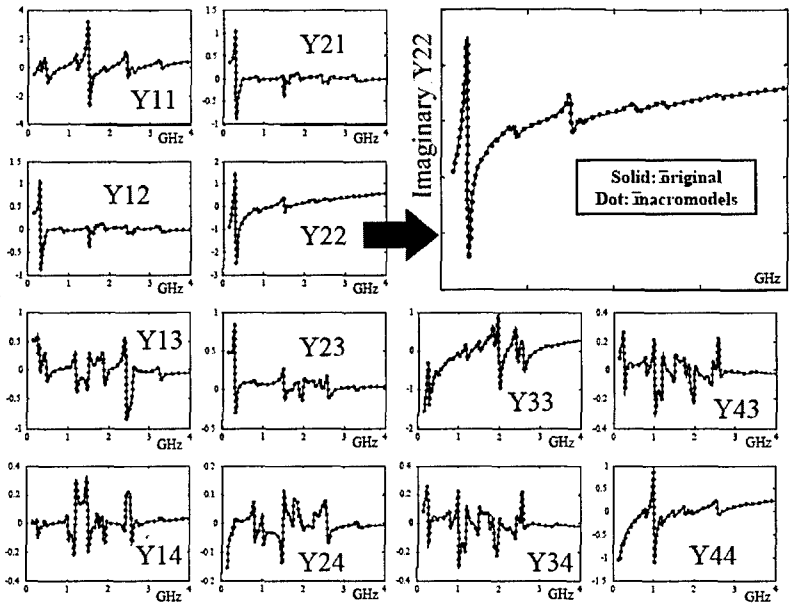


Fig. 3.15 Comparison of imaginary part of admittance parameters for the power plane pair

b) 4-port transmission line data from measurement:

A total of 799 uniformly distributed frequency samples from 50.0 MHz to 20.0 GHz for a 4-port transmission line was measured using a vector network analyzer. The entire frequency band was divided into 16 subbands having a bandwidth of 1.25 GHz. The macromodel was constructed from the lower subband using the orders *NS* of 8 and *DS* of 9 in all subbands. The number of local and global iterations for the pole replacement method was 2 and 200, respectively. The number of complex conjugate poles and real poles extracted were 224 and 1, respectively. Fig.s 3.16 and 3.17 show the comparison between original frequency data and the response of the constructed macromodel for real (Y11) and imaginary (Y12) admittance parameters, respectively.

The macromodel was connected to a 400mV source having 50ps rise/fall times and a 50Ω termination at port 1. Resistors with a 50Ω value were terminated at ports 2, 3, and 4. The time-domain waveform at ports 1 and 2 are shown from 0 to 30ns in Fig. 3.18, demonstrating stability and passivity.

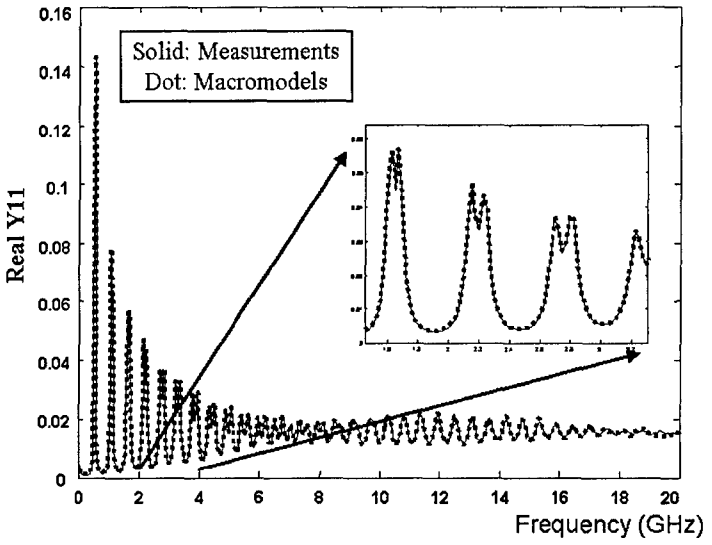


Fig. 3.16 Comparison of real part of admittance parameters

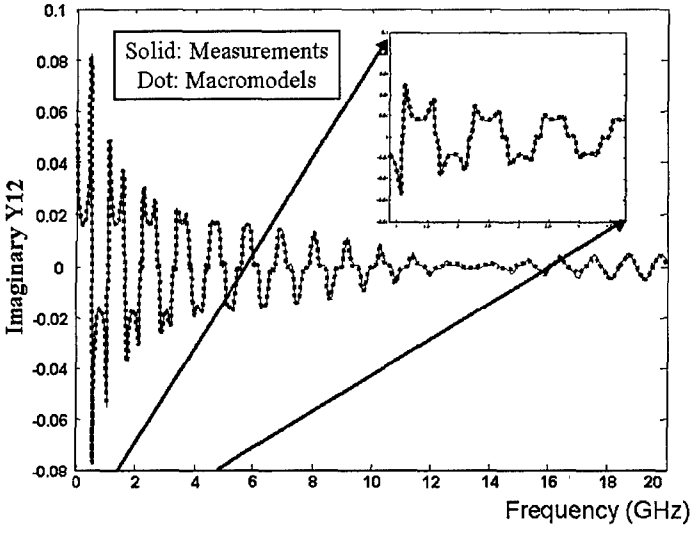


Fig. 3.17 Comparison of imaginary part of admittance parameters

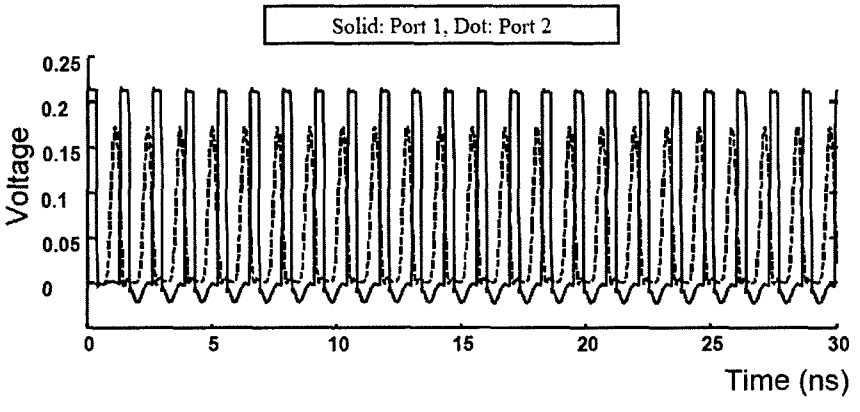


Fig. 3.18 Time domain waveform of the transmission line response

3.4 Summary

BEMP is a software tool for generating macro-models of arbitrary multi-port passive networks, which can be used in a circuit simulator like SPICE. These macro-models accurately capture the electromagnetic effects of the passive networks which are important in the time-domain simulation of digital and mixed-signal systems. The software is coded using the C++ programming language and is executable on the Windows operating system. The program accepts the S, Y or Z parameters of passive networks in the *Touchstone* format. The output is a lumped element sub-circuit file which can be easily integrated into a SPICE simulation. BEMP also generates a macro-model of the input structure represented by rational functions that can be used in frequency domain analysis and simulation. The current version BEMP v3.0 incorporates macro-modeling of broadband multi-port frequency response data, guaranteeing stability and passivity in the generated macro-models. The program has been tested for a variety of systems ranging from coupled transmission lines to irregular shaped power/ground planes to a high-speed data bus. The simulations using the generated macro-models have demonstrated good model accuracy. The program is being further developed to enforce causality on the generated macro-models and include the macro-modeling of nonlinear devices like drivers and receivers. These updates will be incorporated in BEMP v4.0.

References:

1. S. Min and M. Swaminathan, "Efficient Construction of Passive Macro-models for Resonant Networks", Proceedings of the 10th Topical Meeting on Electrical Performance of Electronic Packaging, Boston, Massachusetts, Oct. 2001.
2. S. Min and M. Swaminathan, "Construction of broadband passive macro-models from frequency data for distributed interconnect networks", accepted for publication in the IEEE Transactions on Electromagnetic Compatibility.
3. R. Mandrekar and M. Swaminathan, "Delay Extraction from Frequency Domain Data for Causal Macro-modeling of Passive Networks", Submitted to the International Symposium on Circuits and Systems 2005.
4. K. Judd and A. Mees, "On Selecting Models for Nonlinear Time Series," *Physica D*, Vol. 82, pp. 426–444, 1995.
5. Bhyrav Mutnury, Madhavan Swaminathan and James Libous, "Modeling of Power Supply Noise using Efficient Macro-Model of Non-Linear Driver," *2004 IEEE International Symposium on EMC,* San Jose (CA), USA, August 9-13, 2004.
6. Bhyrav Mutnury, Madhavan Swaminathan and Jim Libous, "Macro-Modeling of Non-Linear Digital I/O Drivers," accepted for *IEEE Transactions on Advanced Packaging*, 2004
7. Bhyrav Mutnury, Madhavan Swaminathan, Moises Cases, Nam Pham, Daniel N. de Araujo and Erdem Matoglu, "Macro-Modeling of Transistor Level Receiver Circuits," accepted at *13th Tropical Meeting on Electrical Performance of Electronic Packaging (EPEP 2004)*, Portland, Oregon, October, 2004

CHAPTER 4 PHYSICS BASED REDUCED ORDER MODELING METHODS

4.1 Mathematical Models

The primary test device for the NeoCAD algorithms developed in this program is the fourth-order continuous time delta-sigma modulator (ΔΣM) [4-1], for which a block diagram is shown in Fig. 4-1. The RF signal enters the modulator on the left at the input gain cell g_0 and the output stream of +1's and –1's is obtained following the third latch at the right. The principle components of the modulator are transconductance amplifiers ($g_{0\text{-}5}$), transimpedance integrators ($1/sC_i$), digital-to-analog converters (DACs) ($g_{6\text{-}9}$), and a voltage amplifier Q_{fb}, 3 latches and an external clock that drives the latches. We have developed behavioral models for each of these components starting from simplified transistor-level models. We have also developed full circuit level models of these components as discussed in previous sections. The mathematical analysis starts from a simple linear model of the ΔΣM with the quantizer approximated by the sampled "sign" function as discussed in section 4.1.2. The output of the modulator, which is a stream of +1 and –1's, is multiplied by a window function (the Kaiser window is used for all our calculations), and the discrete Fourier transform of these numbers gives the modulator spectrum. The signal-to-noise ratio can be obtained directly from the spectrum.

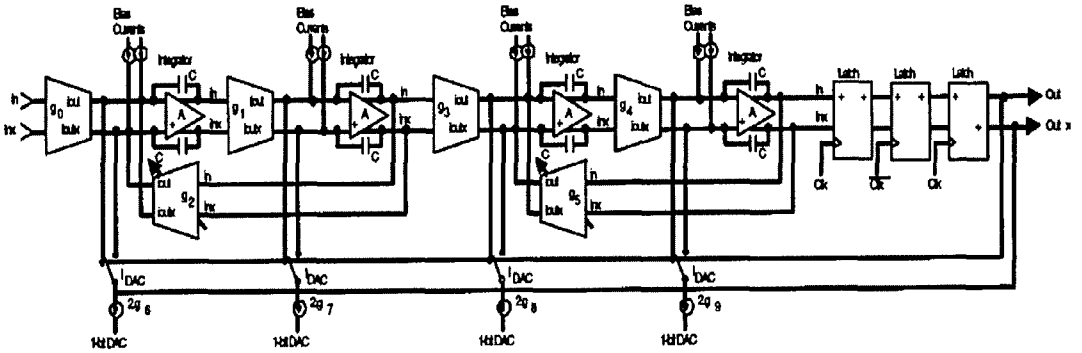


Fig. 4-1. Schematic block diagram of the 4th order continuous time delta sigma modulator.

4.1.1 Idealized Delta-Sigma Modulator Model

The linear model of the 4th order ΔΣM is given by the following set of equations:

$$C_1 \frac{\partial x_1(t)}{\partial t} = g_0\, u(t) - g_2\, x_2(t) - g_6\, y(t-\Delta t) \qquad 4\text{-}1$$

$$C_2 \frac{\partial x_2(t)}{\partial t} = g_1\, x_1(t) - g_7\, y(t-\Delta t) \qquad 4\text{-}2$$

$$C_3 \frac{\partial x_3(t)}{\partial t} = g_3\, x_2(t) - g_5\, x_4(t) - g_8\, y(t-\Delta t) \qquad 4\text{-}3$$

$$C_4 \frac{\partial x_4(t)}{\partial t} = g_4\, x_3(t) - g_9\, y(t - \Delta t) \qquad 4\text{-}4$$

$$y(t) = \mathrm{sgn}[x_4(t_n) + Q_{fb}\, y(t - \Delta t)] \text{ for } t_n \le t < t_{n+1} \qquad 4\text{-}5$$

where x_1 to x_4 are the state variables following each of the 4 integrators, C_1 to C_4 are the capacitances of the integrator, g_1 to g_9 are the coefficients of the gain cells, y is the output of the ΔΣM and Q_{fb} is the quantizer feedback coefficient. The subscript n in the equation for $y(t)$ represents the n^{th} clock period and Δt represents the effective delay of the feedback DAC. Note that eqs. 4-1 to 4-5 are linear between clock periods ($t_n < t < t_{n+1}$).

We have developed a set of parameters that lead to good DSM performance as given in Table 1. The input signal $u(t)$ is given by

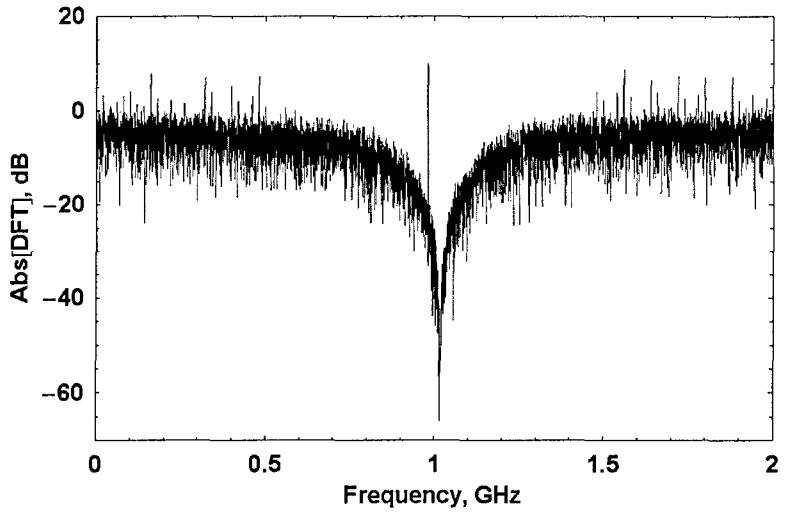
$$u(t) = u_0 \sin(2\pi \nu t). \qquad 4\text{-}6$$

where u_0 is the amplitude and ν is the frequency.

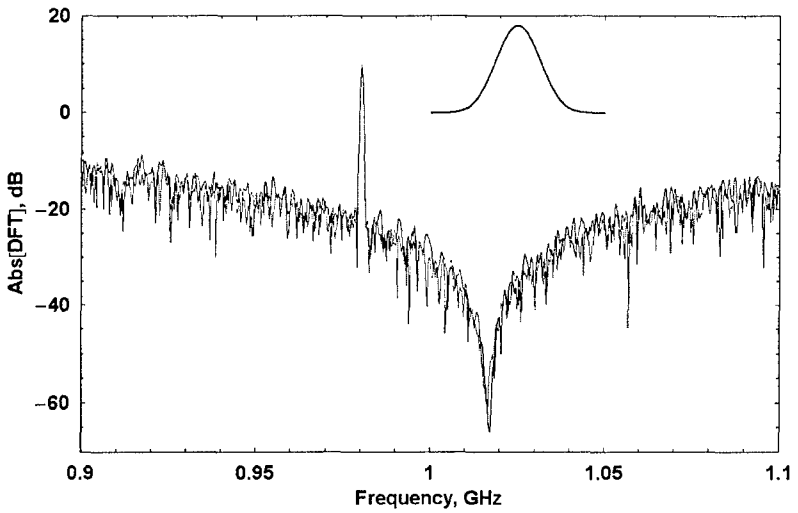
$g_0 = 1.660\ 10^{-3}$	$C_1 = 1.$ pF
$g_1 = 6.383\ 10^{-3}$	$C_2 = 1.$ pF
$g_2 = 6.383\ 10^{-3}$	$C_3 = 1.$ pF
$g_3 = 2.500\ 10^{-3}$	$C_4 = 1.$ pF
$g_4 = 6.383\ 10^{-3}$	$Q_{fb} = 0.1646$
$g_5 = 6.383\ 10^{-3}$	$T_c = 0.25$ ns
$g_6 = -4.833\ 10^{-6}$	$\Delta t = 0.125$ ns
$g_7 = -1.933\ 10^{-3}$	$\nu = 980$ MHz
$g_8 = -3.332\ 10^{-3}$	$u_0 = 0.5$
$g_9 = 8.255\ 10^{-5}$	

Table 4-1. Simulation parameters for 4th order continuous time delta-sigma modulator

Results obtained by solving eqs. 4-1 to 4-5 for input given by eq. (4-6) and parameters given in Table 4-1 are shown in Fig. 4-1 for a 16384-clock period calculation. Fig. 4-1 also shows independent Simulink calculations for the same input and parameters. Note the good agreement between the two results including the deep "notch" at $g_1/(2\pi C_1)$ = 1.016 GHz. The signal peak at 980 MHz is identical in each case. The details of the noise peaks vary slightly from one calculation to another, but the envelopes are in good agreement.



(a)



(b)

Fig. 4-2. Discrete Fourier Transform (dB) of quantizer output multiplied by Kaiser Window as a function of frequency. Simulink calculations are in blue; *Mathematica* calculations are in red. (a) Full range. (b) Blow up of the "notch" with the Kaiser Window shown as an insert.

The signal can be moved through notch by changing the input frequency. The average noise in the notch should remain unchanged as shown in Fig. 4-3.

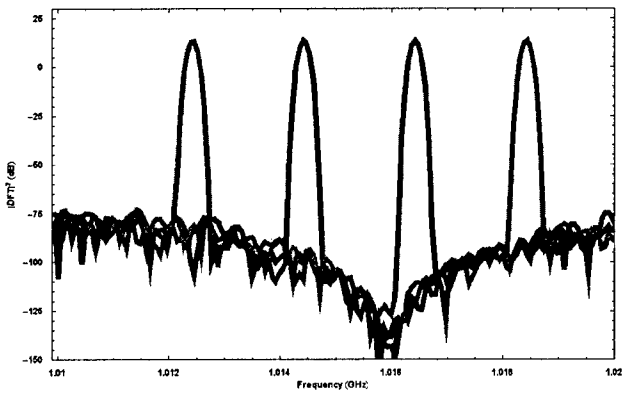


Fig. 4-3. Calculations of ΔΣM spectra for signals with variable input frequency showing that the average ΔΣM noise level is independent of the signal frequency.

The basic gain cell model is shown in Fig. 4-4(a). The input voltage V and output current I are related by

$$V = I R + V_t \ln[(I_{ee}+I)/(I_{ee}-I)] \qquad (4\text{-}7)$$

where I_{ee} is the value of the current source, R is the resistance shown in Fig. 4-4a and V_t is the thermal voltage (~ 0.025 V at room temperature). Fig. 4.4(b) compares the linear approximation to the exact solution of eq. 4-7 for the I-V curve.

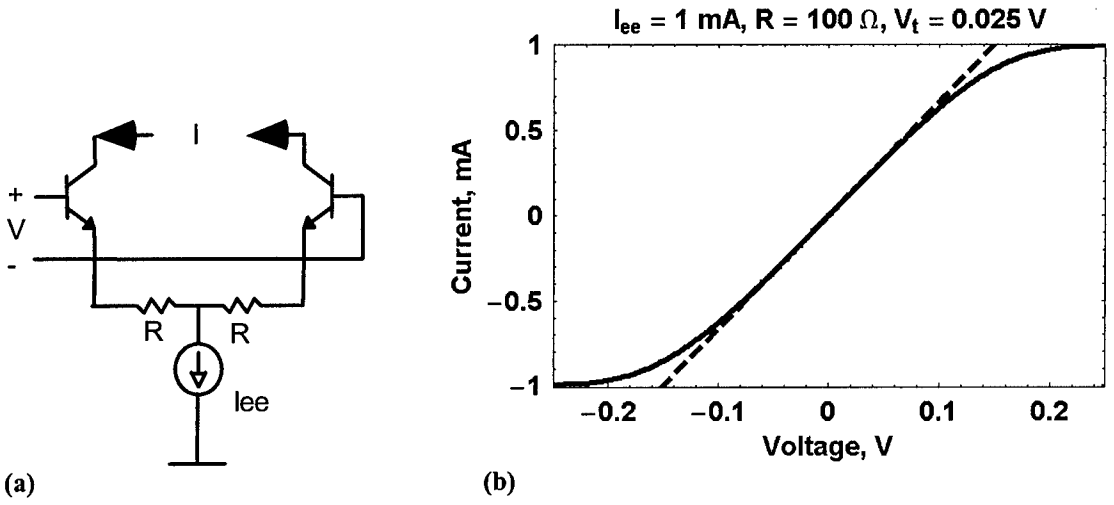


Fig. 4-4. Gain cell model. (a) Circuit schematic. (b) Nonlinear (solid line) and linear approximation (dashed line) to the I-V curve of the gain cell.

Fig. 4-5 shows the effect of gain cell saturation in cell g_1 on ΔΣM SNR as a function of I_{ee}. Note that to keep the small signal gain coefficient at 6.383 10^{-3} as given in Table 4-1, it is necessary to vary R and I_{ee} simultaneously; as can be seen from eq. (4-7), varying I_{ee} alone changes the small signal gain coefficient. The SNR plotted in this and subsequent Fig.s is defined as the ratio of the power in the signal divided by the

integral of the noise spectrum over the specified frequency band centered on the minimum of the notch.

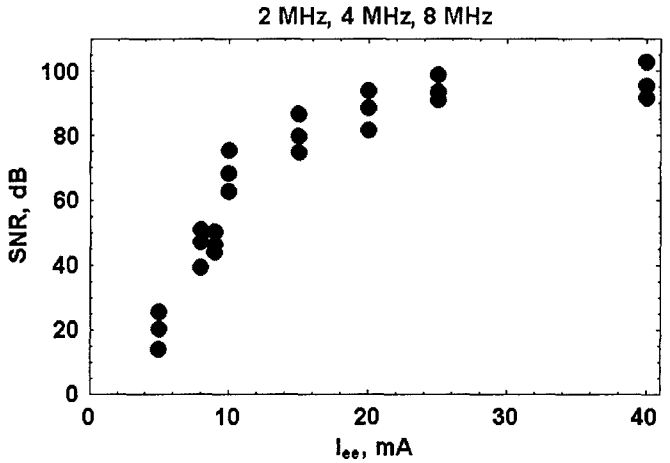


Fig. 4-5. SNR as a function of I_{ee} in gain cell 1 for the baseline 4th order ΔΣM. The SNR is given for 3 values of the noise bandwidth centered about the minimum of the notch.

Gain cell saturation leads to a second deleterious effect, intermodulation products, when the input signal consists of two or more tones. To investigate this effect, we allowed gain cell 0 to saturate and used an input signal having the form of eq. (4-6) plus a second tone 50 dB down in power and 2 MHz lower in frequency than the primary signal. Visualization of spurs in ΔΣM spectra is critically dependent on the number of clock periods in the simulation as shown in Fig. 4-6. For signals separated by 2 MHz, spur resolution requires between 64000 and 260,000 clock-period simulations.

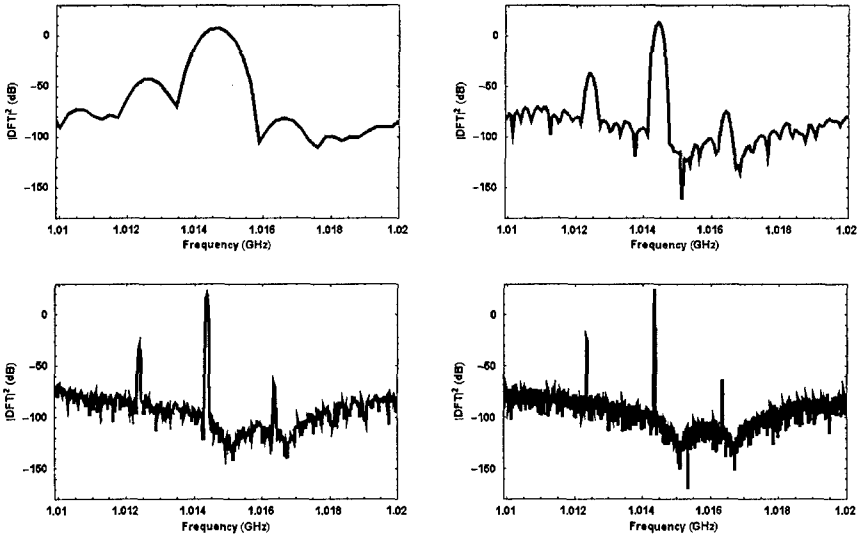


Fig. 4-6. Spur resolution as a function of number clock periods in the simulation: from upper left to lower right: 16384, 65536, 262144, and 1048576 clock periods.

4.1.2 Quantizer

In the previous sections the quantizer was modeled by the sign function as described in eq. (4-5). In this section, we generalize the quantizer behavioral model to a simplified circuit model. In the 4th order ΔΣM shown in Fig. 4-1, the quantizer consists of three latches in series. The operation of each latch is controlled by the output from the clock located elsewhere in the integrated circuit and described in more detail in Section 4.1.5. Here we assume that the clock circuit output is a perfect square-wave voltage signal with period T_c and peak-to-peak amplitude V_{clock}. Mathematically, the clock voltage is then given by

$$\begin{aligned} V_c(t) \quad = \quad & +V_{clock}/2 \text{ for } t_n <= t < t_n + T_c/2 \\ & -V_{clock}/2 \text{ for } t_n + T_c/2 <= t < t_n + T_c = t_{n+1} \end{aligned} \qquad \textbf{(4-8)}$$

Each latch can be modeled as two RC circuits and two gain cells as shown in Fig. 4-6. This circuit operates in two modes that depend on the sign of the clock signal and that we will refer to as "tracking" and "latching". If the clock is positive, the circuit operates in the tracking mode; if negative, it latches. Physically, we can describe the tracking behavior as the situation in which the output voltage of the circuit depends on the input voltage over the entire time interval (half the clock period in this case). While the output in the tracking mode is not an amplified replica of the input, it does follow the input behavior to some extent. In the latching mode the output voltage of the circuit depends only on the input voltage at the beginning of the half clock period—hence the concept of latching onto the signal at a given instant.

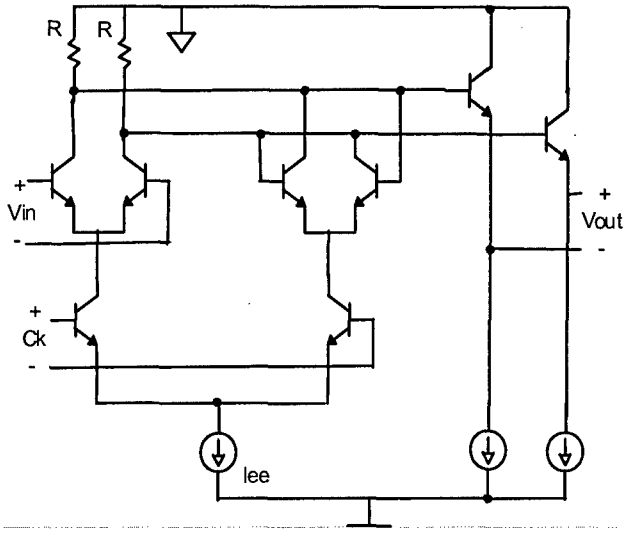


Fig. 4-7. Schematic diagram of the latch circuit.

The output voltage V of the latch circuit shown in Fig. 4-7 can be described by the following differential equation:

$$\tau \frac{dV(t)}{dt} + V(t) = \frac{1}{2} I_{ee} R \tanh\left(\frac{V(t)}{2V_t}\right)\left(1 - \frac{V_C(t)}{V_{clock}/2}\right) + \frac{1}{2} I_{ee} R \tanh\left(\frac{V_0(t)}{2V_t}\right)\left(1 + \frac{V_C(t)}{V_{clock}/2}\right) \tag{4-9}$$

where $V_0(t)$ is the input and $V(t)$ is the output voltage, $\tau = R\,C$ is the RC time constant, I_{ee} is the current produced by the current source, and V_t is the thermal voltage, Boltzman's constant times temperature, $k_B T$. The terms on the left hand side of eq. (4-9) are the standard RC time constant terms. The terms on the right hand side can be derived from the conventional transistor equation with the resistor at the emitter R_e set to zero

$$V = I R_e + V_t \ln[(I_{ee}+I)/(I_{ee}-I)] \tag{4-10}$$

which can be solved for I when $R_e = 0$:

$$I = I_{ee} \tanh(V/V_t) \tag{4-11}$$

The delta sigma modulator shown in Fig. 4-1 uses the three latches at the right as a quantizer; each latch is described by eq. (4-9). The only difference between them is that the latch in the middle is driven by "clock bar" which is defined as the negative of the clock voltage, so that response of the middle latch can be described as latching during the first half of the clock period when the clock is positive and tracking during the second half of the period.

Typical values of the parameters are R = 300 ohms, I_{ee} = 1 mA, V_t = 0.02586V, and C = 30 fF. Before solving for the three-latch system it is useful to look at the properties of eq. (4-9) for positive and negative clock signals. If the clock is negative, the latch mode is operational and eq. (4-9) reduces to

$$\frac{dv(t)}{dt} + v(t) = \beta \tanh[v(t)] \tag{4-12}$$

where $\beta = R\,I_{ee}/V_t$ = 5.8 and t has been normalized to the RC time constant. This equation has the solutions shown in Fig. 4-8 for values of the input voltage V_{in} at t = 0. Note that for $V_{in} << 1$, the solutions to eq. (4-12) require many RC time constants to latch to the input signal. When the latch has a low input signal and a long RC time constant, there may not be enough time for the output to regenerate to a logical level. This causes non-ideal inputs to the DAC, and potentially leads to inconsistencies between the feedback DAC signal and the digitized output value that result in a degraded SNR.

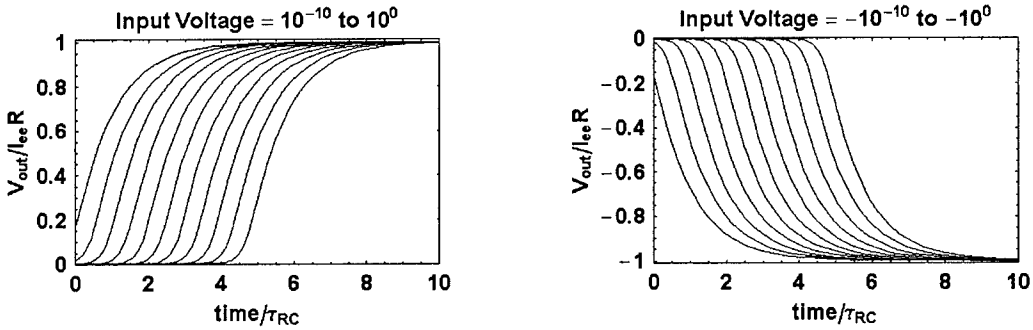


Fig. 4-8. Output voltage as a function of time for the comparator operating in the latch mode.

For a latch in the "following" mode eq. (4-9) reduces to:

$$\frac{dv(t)}{dt} + v(t) = \beta \tanh[\beta V_0(t) / R I_{ee}] \tag{4-13}$$

This equation has the usual quadrature solution with two terms: one corresponds to decay of the initial condition and the second to the response of an RC circuit driven by a possibly saturated gain cell. For $V_0(t)$ sufficiently small the hyperbolic tangent function may be approximated by its argument, and the input voltage drives the RC circuit directly. If the argument of the tanh is large, its value is plus or minus one depending on the sign of $V_0(t)$ and the ouput of the RC circuit reflects little more than the sign of $V_0(t)$.

We have solved the equations for the three-latch system shown in Fig. 4-1 with the results shown in Figs. 4-9 to 4-11. The rows shown in Figs. 4-9 to 4-11 correspond to the input $V_0(t)$, the two internal voltages $V_1(t)$ and $V_2(t)$ and the output $V_3(t)$. The columns correspond to the first 5 clock periods. Note that the input voltage is normalized to its peak value while the internal and output voltages are normalized to $R\,I_{ee} = 0.3$ V.

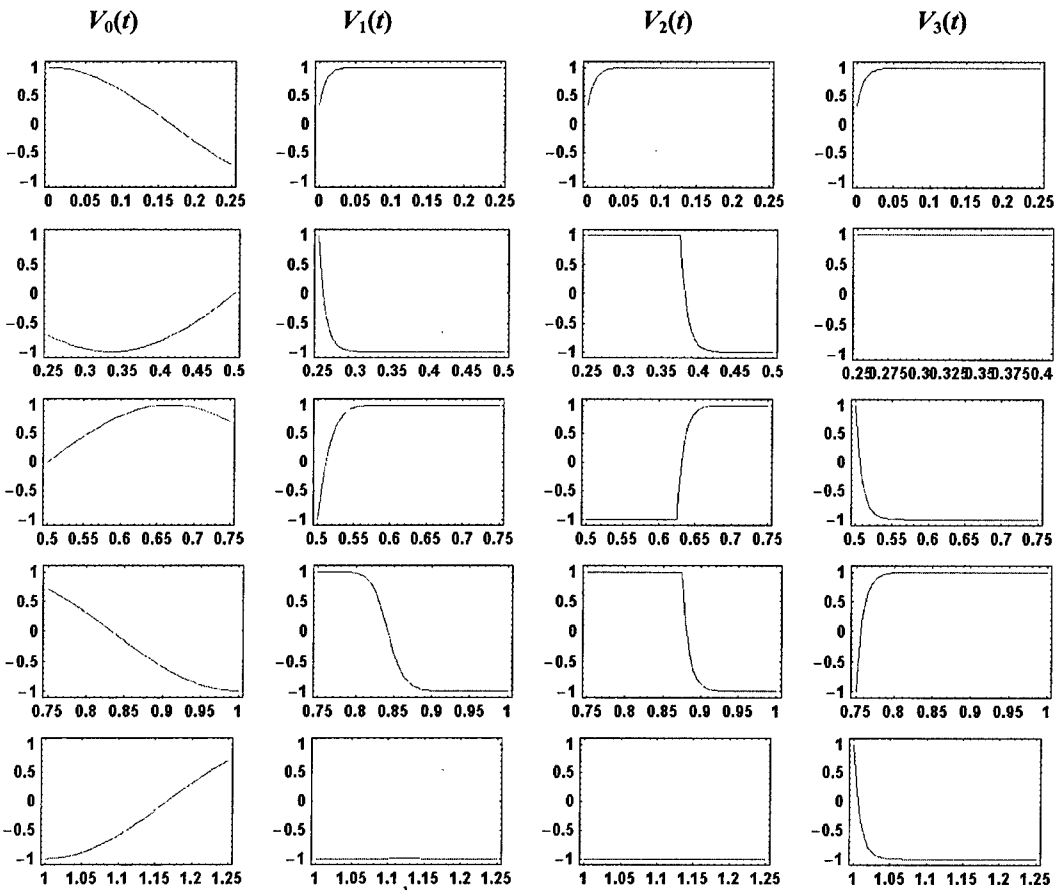


Fig. 4-9. Input voltage $V_0(t)$, the two internal voltages $V_1(t)$ and $V_2(t)$ and the output voltage $V_3(t)$ as a function of time in ns for a clock period of 0.25 ns and an RC time constant of 9 ps. The input voltage to the quantizer is a sine wave 0.6 V peak to peak and a period of 0.667 ns.

Fig. 4-10 illustrates the performance for input voltage = 0.0006 V and the same 9 ps RC time constant. Note that while $V_1(t)$ and $V_2(t)$ are substantially different, the quantized output from the third latch is essentially the same. Fig. 4-11 illustrates the performance for input voltage = 0.0006 V and a 27 ps RC time constant. Now the quantized output contains 2 bit errors in the first 5 bits.

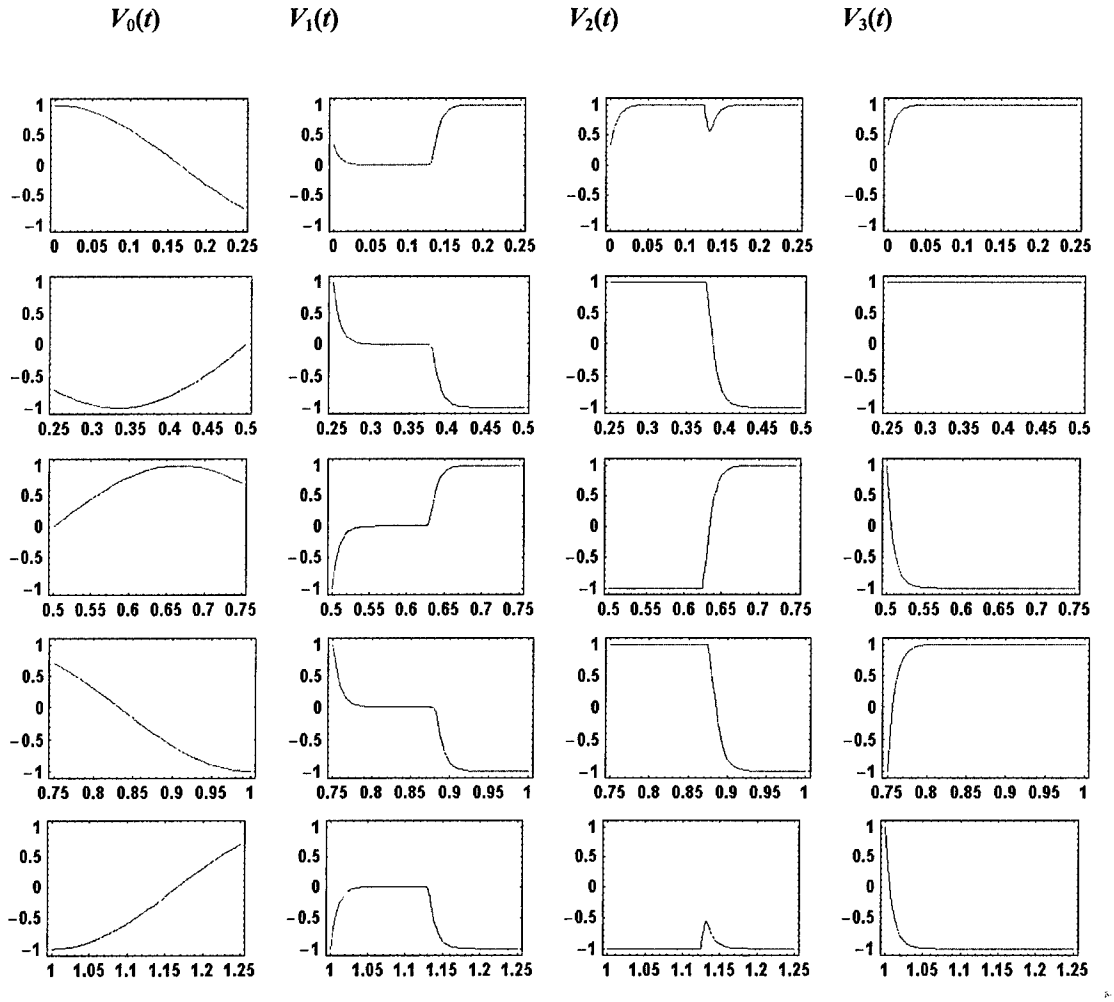


Fig. 4-10. Input voltage $V_0(t)$, the two internal voltages $V_1(t)$ and $V_2(t)$ and the output voltage$V_3(t)$ as a function of time in ns for a clock period of 0.25 ns and an RC time constant of 9 ps. The input voltage to the quantizer is a sine wave 0.0006 V peak to peak and a period of 0.667 ns.

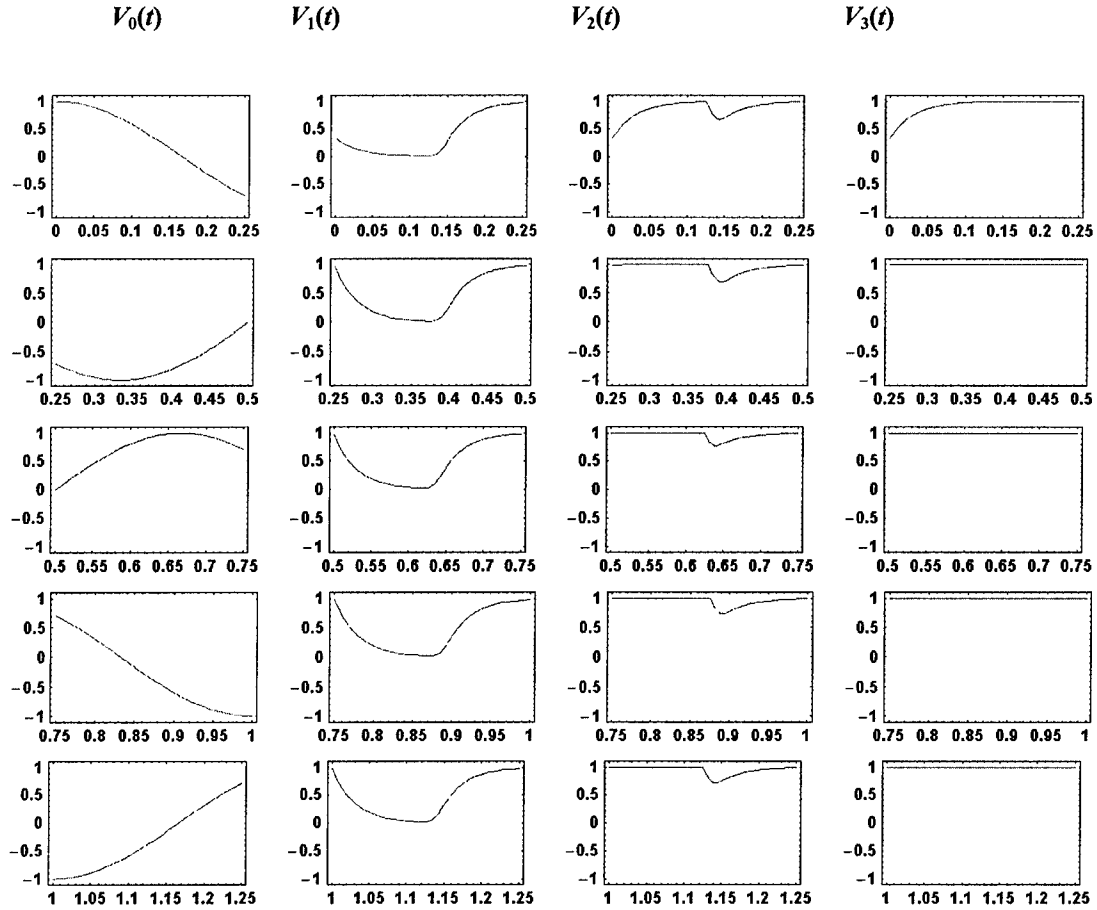


Fig. 4-11. Input voltage $V_0(t)$, the two internal voltages $V_1(t)$ and $V_2(t)$ and the output voltage $V_3(t)$ as a function of time in ns for a clock period of 0.25 ns and an RC time constant of 27 ps. The input voltage to the quantizer is a sine wave 0.0006 V peak to peak and a period of 0.667 ns.

The behavioral model of the quantizer has been used in place of eq. (4-5) to simulate the ΔΣM with the results shown in Fig. 4-12. Note that the noise level in the blue curve for the circuit level model is slightly increased from the more ideal calculation in red based on eqs. (4-1) to (4-5). The signal is identical in each case and the location of the notch is the same to within simulation error.

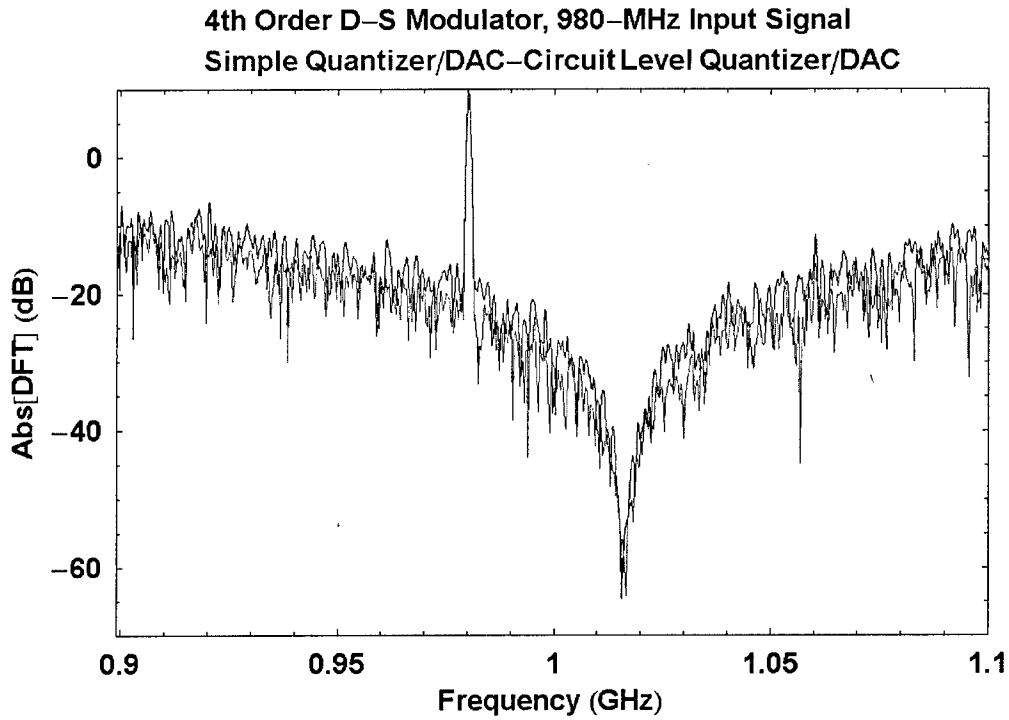


Fig. 4-12 Absolute value of the discrete Fourier transform of the output of the fourth order delta sigma modulator times the Kaiser window as a function of frequency in GHz.

4.1.3 Clock Model

The square wave clock model has been revised to include the finite rise time as shown by Fig. 4-13. This clock function is used in the complete ΔΣM module discussed in Section 4.1.8. For the rise time shown in Fig. 4-13, approximately 0.02 ns, no significant effect on ΔΣM noise shaping is observed as shown in Fig. 4-14 which compares the spectra with the square wave clock to that with the clock waveform shown in Fig. 4-13.

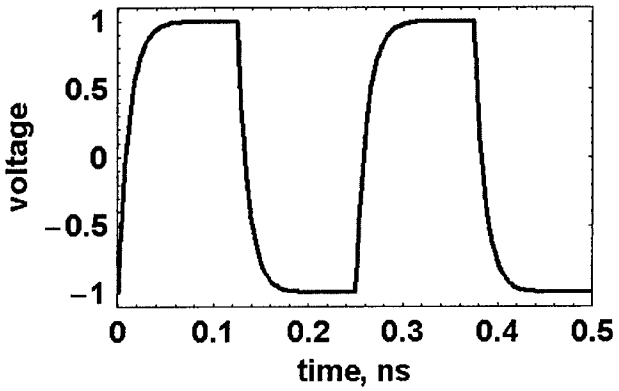


Fig. 4-13. Clock voltage as a function of time with finite rise time.

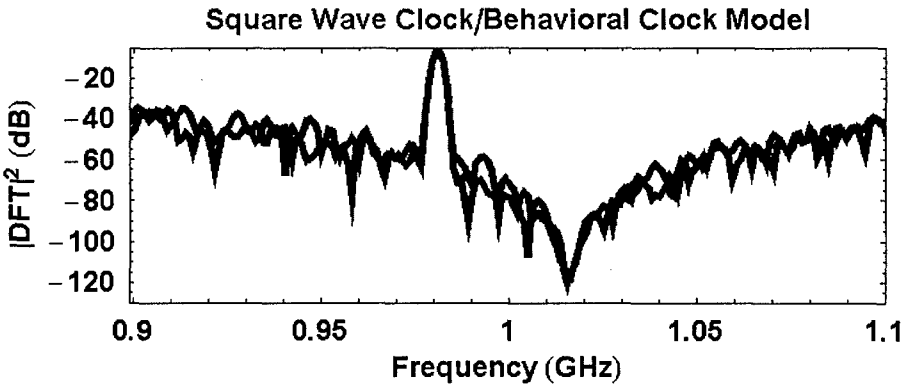


Fig. 4-14. High resolution DSM spectra for a square-wave clock and the behavioral model of the clock shown in Fig. 4-13.

4.1.4 Digital to Analog Converter

In the simple linear model of the DSM given in Section 4.1.2, the DAC is essentially perfect and the output of the "sign" quantizer is fed back directly into the DSM. A two-transistor model of the DAC is shown in Fig. 4-15. Here the input voltage is V_{in} and the output current I_{out}.

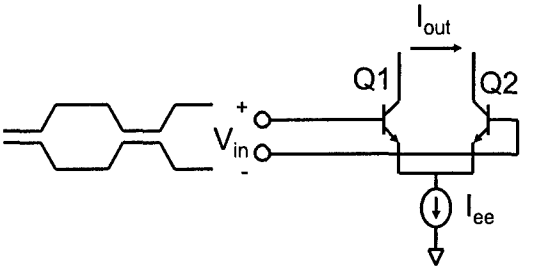


Fig. 4-15: Two-transistor model of DAC.

The equations that characterize this device are

$$V_{in} - V_{be1} + V_{be2} = 0 \qquad (4\text{-}18)$$

$$V_{be1} = V_{t1} \ln(I_{c1}/I_s) \qquad (4\text{-}19)$$

$$V_{be2} = V_{t2} \ln(I_{c2}/I_s) \qquad (4\text{-}20)$$

$$I_{c1} + I_{c2} = I_{tail} \qquad (4\text{-}21)$$

$$I_{out} = (I_{c1} - I_{c2}). \qquad (4\text{-}22)$$

V_{be1} and V_{be2} are the base-emitter voltages on the two transistors, I_{c1} and I_{c2} are the collector currents, V_{t1} and V_{t2} are the thermal voltages at temperature T_1 and T_2, I_{tail} and

I_s are the tail and saturation currents. For $V_{t1} = V_{t2}$, the equations have the simple solution

$$I_{out} = I_{ee} \tanh (V_{in}/2V_t). \tag{4-23}$$

This equation has been used to model the DACs when differential heating (unequal $V_{t1} = V_{t2}$) is ignored. It leads to little change in the DSM behavior because $V_{in} >> 2V_t$, and the tanh can generally be replaced by 1 or –1, which is the approximation used in eqs. (4-1) to (4-5).

Unfortunately, the input voltages to the two transistors are not equal, but depend on the bit stream out of the ΔΣM. Thus a large number of transitionless bits in a row can lead to relatively large differential heating in the two transistors. If T_H and T_L are the equilibrium temperatures when the transistors are on and off, respectively, $T_1(t)$ and $T_2(t)$ are the temperatures of Q1 and Q2 as a function of time, and τ_{th} is the thermal time constant, the temperatures evolve in time according to the following equations.

$$\tau_{th} \frac{\partial T_1(t)}{\partial t} = \Delta T\, y(t) + T_0 - T_1(t) \tag{4-24}$$

$$\tau_{th} \frac{\partial T_2(t)}{\partial t} = -\Delta T\, y(t) + T_0 - T_2(t) \tag{4-25}$$

where $T_0 = (T_H + T_L)/2$ and let $\Delta T = (T_H - T_L)/2$. With unequal temperatures, the simple analytical solution given by eq. (4-23) is lost and the equations become a little more difficult. It can be shown that for sufficiently small differences in the temperatures of the two transistors eq. (4-23) becomes

$$I_{out} = I_{ee} \tanh [V_{in}/2V_t - c_{th} (T_1 - T_2)/T_0] \tag{4-26}$$

where c_{th} is a constant that depends on the device parameters. In general, there is other temperature dependence in the device models of the two transistors that lead to a similar form of eq. (4-26) so c_{th} is best regarded as an empirical constant, which for our devices is approximately 6. Eqs. (4-24) to (4-26) and the three-latch model of the quantizer have been solved simultaneously with eqs. (4-1) to (4-4). Fig. 4-16 shows SNR as a function of ΔT for $\tau_{th} = 1$ ns. Fig. 4-17 shows SNR as a function of the thermal time constant with ΔT = 40K.

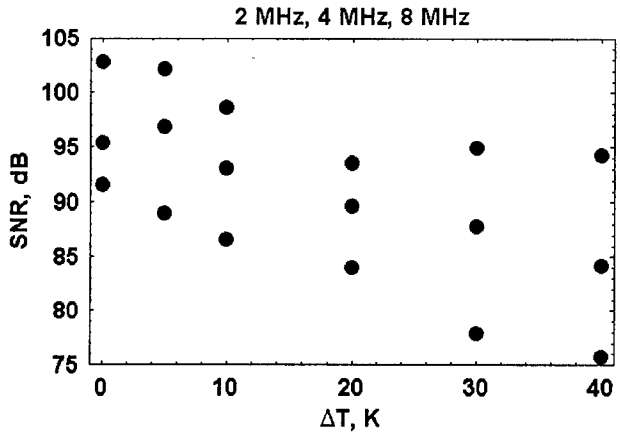


Fig. 4-16. SNR as a function of ΔT for a thermal time constant of τ_{th} = 1 ns.

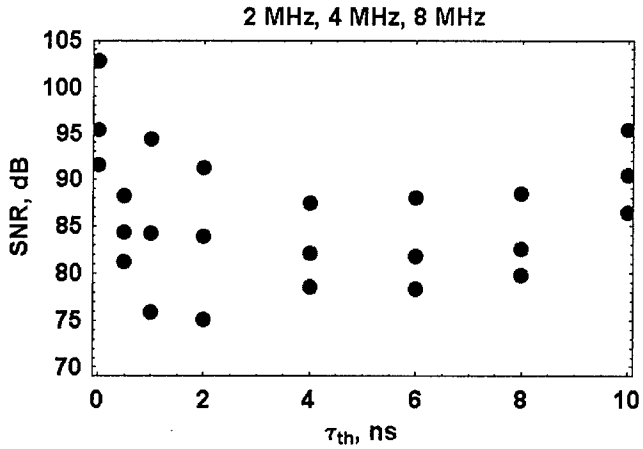


Fig. 4-17. SNR as a function of τ_{th} for ΔT = 40K.

4.1.5 Integrator

Fig. 4-18 shows an implementation of a simple integrator [4-1]. It is designed so that the output voltage is ideally proportional to the integral of the input current, so that

$$V_{out} = \frac{1}{C}\int I_{in}\,dt \tag{4-27}$$

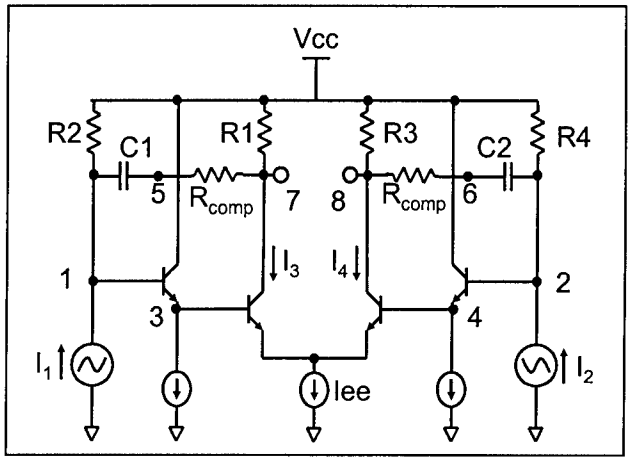


Fig. 4.18 : Simple Integrator

The circuit is fully differential, the output voltage is given by $V_{out} = V_8 - V_7$, and the input current is defined as $I_{in} = I_1 - I_2$. By the definition of V_{out}, the integrator is noninverting. R_{comp} compensates for the limited value of I_{ee} by bringing the pole closer to zero.

We can then solve for the input-output relationships for this circuit by using Kirchoff's Current Laws at each node. Assuming $\boldsymbol{R_1 = R_2}$, $\boldsymbol{R_3 = R_4}$, and $\boldsymbol{C_1 = C_2 = C}$, we find:

$$\text{Node 1: } I_1 + C\frac{d}{dt}(V_5 - V_1) + \frac{V_{cc} - V_1}{R_2} = 0 \tag{4-28}$$

$$\text{Node 2: } I_2 + C\frac{d}{dt}(V_6 - V_2) + \frac{V_{cc} - V_2}{R_2} = 0 \tag{4-29}$$

$$\text{Node 5: } C\frac{d}{dt}(V_1 - V_5) + \frac{V_7 - V_5}{R_{comp}} = 0 \tag{4-30}$$

$$\text{Node 6: } C\frac{d}{dt}(V_2 - V_6) + \frac{V_8 - V_6}{R_{comp}} = 0 \tag{4-31}$$

$$\text{Node 7: } I_3 + \frac{V_7 - V_5}{R_{comp}} + \frac{V_7 - V_{ee}}{R_1} = 0 \tag{4-32}$$

$$\text{Node 8: } I_4 + \frac{V_8 - V_6}{R_{comp}} + \frac{V_8 - V_{ee}}{R_1} = 0 \tag{4-33}$$

From eq. (4-23), we obtain

$$I_3 - I_4 = I_{ee} \tanh\left(\frac{V_3 - V_4}{2V_t}\right) \tag{4-34}$$

We then assume perfect emitter-followers so that

$$V_3 - V_4 = V_1 - V_2 \tag{4-35}$$

This gives us the necessary equations. However, they can be simplified for faster modeling. Subtracting (4-29) from (4-28),

$$I_1 - I_2 + C\frac{d}{dt}(V_5 - V_6) - C\frac{d}{dt}(V_1 - V_2) - \frac{V_1 - V_2}{R_2} = 0 \tag{4-36}$$

Subtracting (4-31) from (4-30),

$$C\frac{d}{dt}(V_1 - V_2) - C\frac{d}{dt}(V_5 - V_6) + \frac{V_7 - V_8}{R_{comp}} - \frac{V_5 - V_6}{R_{comp}} = 0 \tag{4-37}$$

Subtracting (4-33) from (4-32),

$$I_3 - I_4 + (V_7 - V_8)\left(\frac{1}{R_{comp}} + \frac{1}{R_1}\right) - \frac{V_5 - V_6}{R_{comp}} = 0 \tag{4-38}$$

and using the expressions in (4-34) and (4-35) for $\boldsymbol{I_3 - I_4}$, we obtain

$$I_{ee}\tanh\left(\frac{V_1 - V_2}{2V_t}\right) + (V_7 - V_8)\left(\frac{1}{R_{comp}} + \frac{1}{R_1}\right) - \frac{V_5 - V_6}{R_{comp}} = 0 \tag{4-39}$$

We now simplify, recognizing we are only concerned with differential signals. We let

$$\boldsymbol{I_{in} = I_1 - I_2} \tag{4-40}$$

$$\boldsymbol{V_{out} = V_8 - V_7} \tag{4-41}$$

$$\boldsymbol{V_a = V_1 - V_2} \tag{4-42}$$

$$\boldsymbol{V_b = V_5 - V_6} \tag{4-43}$$

which then yields

$$I_{in} + C\frac{dV_b}{dt} - C\frac{dV_a}{dt} - \frac{V_a}{R_2} = 0 \tag{4-44}$$

$$C\frac{dV_a}{dt} - C\frac{dV_b}{dt} - \frac{V_{out}}{R_{comp}} - \frac{V_b}{R_{comp}} = 0 \tag{4-45}$$

$$I_{ee}\tanh\left(\frac{V_a}{2V_t}\right) - V_{out}\left(\frac{1}{R_{comp}} + \frac{1}{R_1}\right) - \frac{V_b}{R_{comp}} = 0\,. \tag{4-46}$$

We can then describe the input-output relationship using eqs. (4-44) - (4-46). Some practical values for this circuit are $\boldsymbol{C_1 = C_2}$ = 1 pF, $\boldsymbol{R_1 = R_3}$ = 500 Ω, $\boldsymbol{R_2 = R_4}$ = 10 kΩ, and $\boldsymbol{I_{ee}}$ = 20 mA. $\boldsymbol{R_{comp} = 2V_t/I_{ee}}$ = 2.586 Ω.

This integrator was placed into an otherwise ideal behavioral model in Verilog-A using the standard NeoCAD coefficients. With an ideal integrator, a 4K FFT gives the spectrum shown in Fig. 4-19.

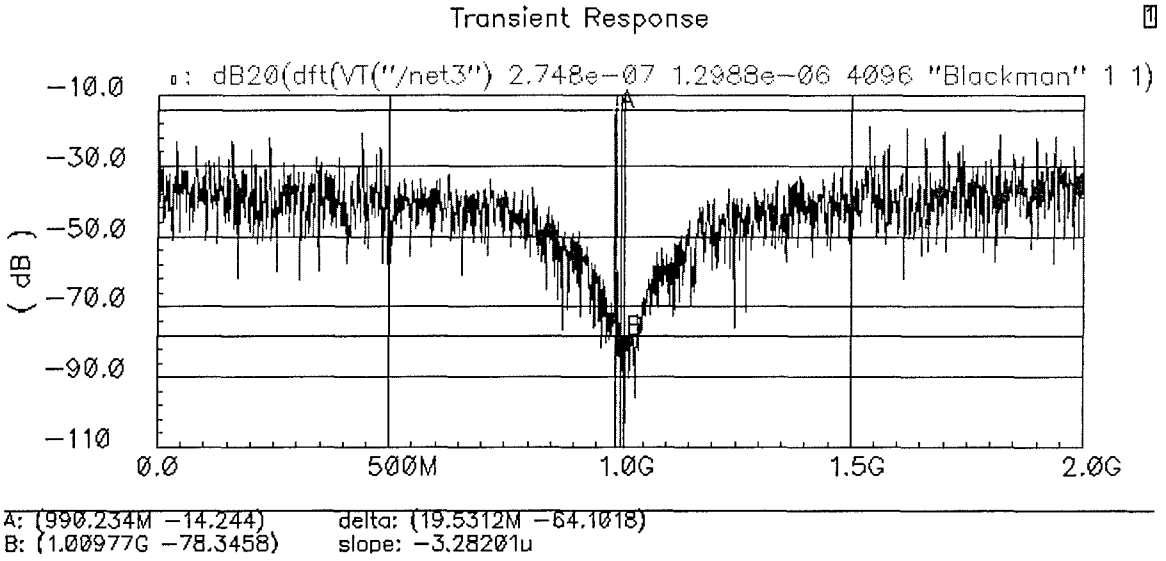


Fig. 4-19. 4K FFT of ideal behavioral-level ΔΣ modulator in Verilog-A.

The SNR is about 64 dB in 1 MHz bandwidth. With the integrator model in this memo, the 4K FFT gives the spectrum shown in Fig. 4-20.

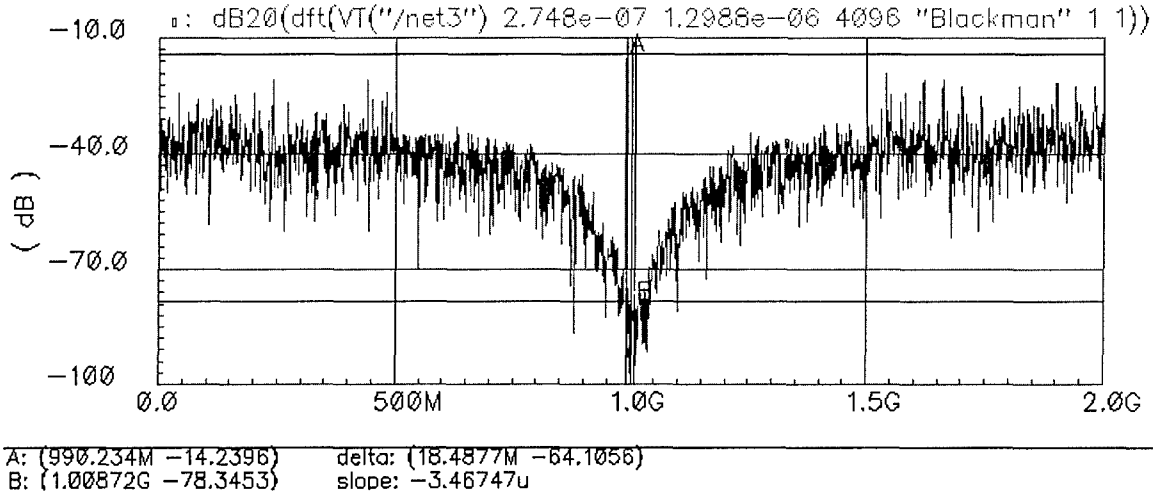


Fig. 4-20. 4K FFT of ideal behavioral-level ΔΣ modulator in Verilog-A, with the exception of the integrator model given by Eqs. (4-44)-(4-46).

The SNR is now approximately 64 dB in 1 MHz BW, which gives very similar results to that of an ideal integrator.

The integrator model given by eqs. (4-44) and (4-46) has been added to the ΔΣM behavioral simulation including the other effects discussed in Sections 4.1.3 to 4.1.6. When the current I_{ee} is sufficiently large, ideal performance equivalent to solutions of eqs. (4-1) to (4-5) can be recovered. As the current is reduced the noise shaping of the ΔΣM becomes less effective and the SNR is reduced as shown in Fig. 4-21. Note that for a noise bandwidth of 8 MHz, little is gained by increasing the integrator current above approximately 30 mA.

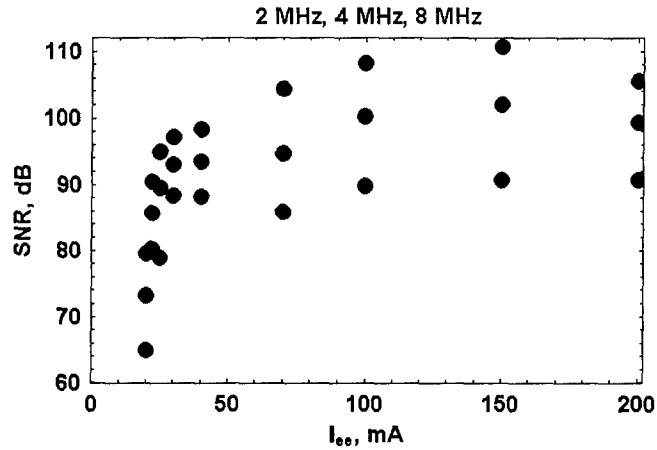


Fig. 4-19. ΔΣM SNR as a function of the integrator current I_{ee}.

To illustrate the use of the behavioral model of the integrator, we varied $\boldsymbol{R}_{\mathbf{comp}}$ in units of $\boldsymbol{V}_t/\boldsymbol{I}_{ee}$ with the results shown in Fig. 4-20. Note that for $\boldsymbol{R}_{\mathbf{comp}}/(\boldsymbol{V}_t/\boldsymbol{I}_{ee})$ less than about 1.25 the integrator SNR drops rapidly as state variables become unstable. On the other hand for $\boldsymbol{R}_{\mathbf{comp}}/(\boldsymbol{V}_t/\boldsymbol{I}_{ee})$ greater than about 3 the SNR values are not representative of the modulator performance since the spectra clearly show that the modulator has turned into a resonator. The operating regime for the ΔΣM is about $1.25 < \boldsymbol{R}_{\mathbf{comp}}/(\boldsymbol{V}_t/\boldsymbol{I}_{ee}) < 3.25$.

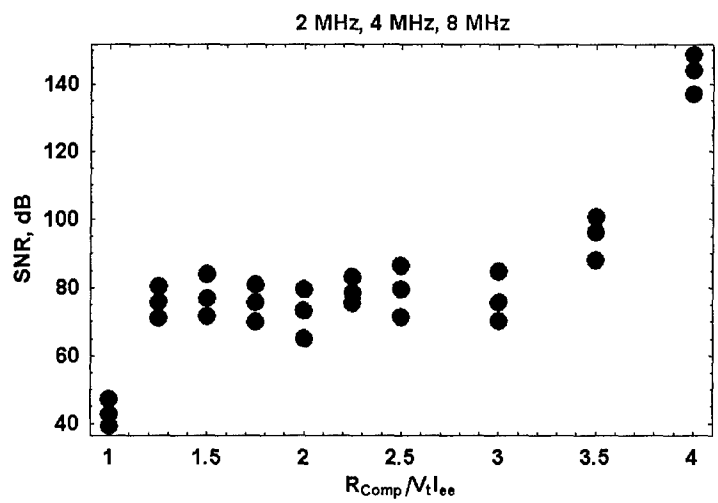


Fig. 4-20. ΔΣM SNR as a function of the integrator current Rcomp/VtI_{ee}.

4.1.6 Full Model: Non-Idealized Delta-Sigma Modulator

The full set of equations for the ΔΣM developed in the previous section are given by

$$I_{ee}\tanh\left(\frac{V_{a1}(t)}{2V_t}\right)-x_1(t)\left(\frac{1}{R_{comp}}+\frac{1}{R_1}\right)-\frac{V_{b1}(t)}{R_{comp}}=0 \tag{4-47}$$

$$-g_6\tanh\left[\frac{V_3(t)}{2V_t}-c_{th}\left(\frac{T_1(t)}{T_0}-1\right)\right]+g_0[u(t)]-g_2[x_2(t)]+C\frac{\partial[V_{b1}(t)-V_{a1}(t)]}{\partial t}-\frac{V_{a1}(t)}{R_2}=0 \tag{4-48}$$

$$-C\frac{\partial[V_{b1}(t)-V_{a1}(t)]}{\partial t}-\frac{x_1(t)}{R_{comp}}-\frac{V_{b1}(t)}{R_{comp}}=0 \tag{4-49}$$

$$I_{ee}\tanh\left(\frac{V_{a2}(t)}{2V_t}\right)-x_2(t)\left(\frac{1}{R_{comp}}+\frac{1}{R_1}\right)-\frac{V_{b2}(t)}{R_{comp}}=0 \tag{4-50}$$

$$g_1[x_1(t)]-g_7\tanh\left[\frac{V_3(t)}{2V_t}-c_{th}\left(\frac{T_1(t)}{T_0}-1\right)\right]+C\frac{\partial[V_{b2}(t)-V_{a2}(t)]}{\partial t}-\frac{V_{a2}(t)}{R_2}=0 \tag{4-51}$$

$$-C\frac{\partial[V_{b2}(t)-V_{a2}(t)]}{\partial t}-\frac{x_2(t)}{R_{comp}}-\frac{V_{b2}(t)}{R_{comp}}=0 \tag{4-52}$$

$$I_{ee}\tanh\left(\frac{V_{a3}(t)}{2V_t}\right)-x_3(t)\left(\frac{1}{R_{comp}}+\frac{1}{R_1}\right)-\frac{V_{b3}(t)}{R_{comp}}=0 \tag{4-53}$$

$$-g_8\tanh\left[\frac{V_3(t)}{2V_t}-c_{th}\left(\frac{T_1(t)}{T_0}-1\right)\right]+g_3[x_2(t)]-g_5[x_4(t)]+C\frac{\partial[V_{b3}(t)-V_{a3}(t)]}{\partial t}-\frac{V_{a3}(t)}{R_2}=0 \tag{4-54}$$

$$-C\frac{\partial[V_{b3}(t)-V_{a3}(t)]}{\partial t}-\frac{x_3(t)}{R_{comp}}-\frac{V_{b3}(t)}{R_{comp}}=0 \tag{4-55}$$

$$I_{ee}\tanh\left(\frac{V_{a4}(t)}{2V_t}\right)-x_4(t)\left(\frac{1}{R_{comp}}+\frac{1}{R_1}\right)-\frac{V_{b4}(t)}{R_{comp}}=0 \tag{4-56}$$

$$-g_9\tanh\left[\frac{V_3(t)}{2V_t}-c_{th}\left(\frac{T_1(t)}{T_0}-1\right)\right]+g_4[x_3(t)]+C\frac{\partial[V_{b4}(t)-V_{a4}(t)]}{\partial t}-\frac{V_{a4}(t)}{R_2}=0 \tag{4-57}$$

$$-C\frac{\partial[V_{b4}(t)-V_{a4}(t)]}{\partial t}-\frac{x_{14}(t)}{R_{comp}}-\frac{V_{b4}(t)}{R_{comp}}=0 \tag{4-58}$$

$$\tau \frac{\partial V_1(t)}{\partial t} + V_1(t) - \left\{ \frac{1}{2} [1 + V_C(t)] V_{sat}\left[Q_{fb} \tanh\left(\frac{V_3(t)}{2 V_t} \right) + x_4(t) \right] + \frac{1}{2} [1 - V_C(t)] V_{sat}[V_1(t)] \right\} = 0 \quad (4\text{-}59)$$

$$\tau \frac{\partial V_2(t)}{\partial t} + V_2(t) - \left\{ \frac{1}{2} [1 - V_C(t)] V_{sat}[V_1(t)] + \frac{1}{2} [1 + V_C(t)] V_{sat}[V_2(t)] \right\} = 0 \quad (4\text{-}60)$$

$$\tau \frac{\partial V_3(t)}{\partial t} + V_3(t) - \left\{ \frac{1}{2} [1 + V_C(t)] V_{sat}[V_2(t)] + \frac{1}{2} [1 - V_C(t)] V_{sat}[V_3(t)] \right\} = 0 \quad (4\text{-}61)$$

$$\frac{\partial T_1(t)}{\partial t} + \frac{T_1(t)}{\tau_{th}} = \frac{T_0 + \Delta T\, V_3(t)}{\tau_{th}} \quad (4\text{-}62)$$

where $V_C(t)$ is the clock function given by an interpolation function for the data plotted in Fig. 4-13 and

$$V_{sat}(V) = R_{Latch}\, I_{eeLatch} \tanh\left(\frac{V}{2 V_t} \right)$$

Use of the parameters R_{Latch} and $I_{eeLatch}$ for the resistance and current of the latches used in the quantizer allows the values of these parameters to be different from those of the same parameters of the integrator.

The first 12 equations in the set (4-47) to (4-62) are obtained by repetitive application of eqs. (4-44) to (4-46) for the four integrators shown in Fig. 4-1 that were represented by the simple time derivatives in eqs. (4-1) to (4-4). The DACs are represented by the $\tanh\{V_3(t)/2V_t - c_{th}[T_1(t)/T_0 - 1]\}$ terms in eqs. (4-48), (4-51), (4-54), and (4-55). Note that it is not necessary to solve for both $T_1(t)$ and $T_2(t)$ because they are antisymmetric for identical transistors in the DAC. Saturation of the gain cells is included by the replacement $g_i\, x_i(t)$ with $g_i[x_i(t)]$ where $g_i[\ldots]$ indicates the inverse function obtained by solving eq. (4-7) [this must be done numerically with an interpolation function since there is no analytical solution for eq. (4-7)] . The quantizer is represented by eqs. (4-59) to (4-61) that consist of 3 versions of eq. (4-9) applied to the three latches in series in Fig. 4-1. Note the middle latch is clocked by "clock bar", which is why the signs in the $[1 - V_C(t)]$ and $[1 + V_C(t)]$ terms are reversed in eq. (4-60). Equstions (4-47) to (4-62) are a set of coupled algebraic and differential equations. Because they contain time constants that vary by orders of magnitude, they are also stiff differential equations. Thus application of a simple differential equation solver such as first order Runge-Kutta is not likely to be successful. This is one of the underlying rationales for the development of HAARSPICE.

4.1.7 Conclusions

We have developed a detailed behavioral level model for the 4th order, continuous time delta sigma modulator that is based on simple transistor-level approximations to the

key components of the modulator: Integrator, Gain Cell, Clock, DAC and Quantizer. Numerical solutions for the equations that describe this model have been performed to obtain frequency spectra for the ΔΣM. These spectra have been carefully benchmarked against circuit-level simulations and Simulink calculations that have been used in the design process at HRL for many years. Use of these behavioral models has enabled us to predict ΔΣM performance as a function of circuit level parameters such as the integrator current and compensation resistance, the gain cell current and resistance, the latch RC time constants, and the thermal response time for transistor heating, and these calculations typically take less than 10 minutes on a 1-GHz PC.

References

[4-1] G. Raghavan, J. F. Jensen, J. Laskowski, M. Kardos, M. G. Case, M. Sokolich, and S. Thomas III, "Architecture, design, and test of continuous-time tunable intermediate-frequency bandpass delta-sigma modulators," IEEE J. Solid-State Circuits, vol. 36, no. 1, Jan. 2001, pp. 5- 13.

[4-2] J.F. Jensen, G. Raghavan, A.E. Cosand, R.H. Walden, "A 3.2-GHz Second-Order Delta-Sigma Modulator Implemented in InP HBT Technology," *IEEE Journal of Solid-State Circuits*, Vol. 30, No. 10, October 1995.

4.2 Hardware Description Langauge (HDL) Models

Hardware description language models (HDLs) exist to describe hardware. In this they differ from traditional programming languages, which generally exist to describe algorithms. Hardware description languages have two primary applications: simulations and synthesis. With simulation, one applies various stimuli to an executable model that is described using the HDL in order to predict how it will respond. Simulation allows you to understand how complex systems behave before you incur the time and expense of implementing them. Synthesis is the process of actually implementing the hardware. In this chapter, Verilog-A models used in sigma-delta modulators will be described. In Section 4.2.1 we describe the ideal model of a 2nd order continuous time delta-sigma modulator using linear electronic components as building blocks. In section 4.2.2 we introduce non-idealities in the delta-sigma modulator and show gradual changes in the performance of the modulator. Finally we summarize the results in section 4.2.3

4.2.1 Ideal Models

Most mixed signal systems and sub-systems can be built using reusable subblocks that can be configured in multiple ways to meet design specifications. We begin with the ideal model of a 2nd order delta-sigma modulator as shown in Fig. 1

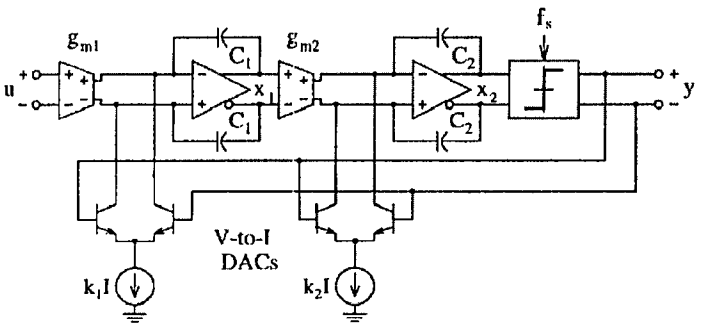


Fig 1: 2nd Order Continuous Time Σ modulator

An ideal 2nd order continuous time delta-sigma modulator can be modeled as a linear circuit in which the quantizer is replaced by the "sign" function. However, the Signal to Quantization Noise (SQNR) is a random function of the input amplitude and frequency, and therefore, the modulator dynamics are not as simple as the linear model predicts. We begin with using the *"sign"* function for the bipolar quantizer, which gives an output of ±1. The other sub-blocks, the Gm-cell and integrator used in the loop filter and the current steering digital-to-analog converter (DAC) are all modeled using ideal components. To simulate the idealized delta-sigma modulator, we have used HAARSPICE and Cadence's Spectre circuit simulators. Currently HAARSPICE accepts Cadence's spectre netlist but cannot parse verilog-A based models. Fig. 2 shows the idealized modulator with linear components. Table 1, given below, lists the one-to-one mapping between the blocks and the components.

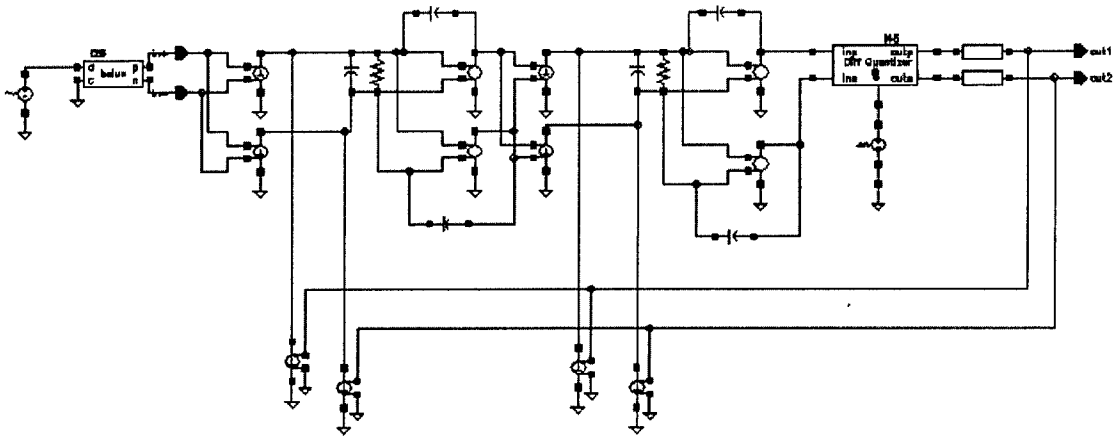


Fig. 2: Idealized 2nd order continuous time delta sigma modulator using electrical components

Delta Sigma Blocks	**Equivalent Circuit Components**
Ideal Gm-Cell	Voltage controlled current source
Ideal Integrator:	Opamp: voltage controlled voltage source Ideal capacitor
Digital to analog converter	Voltage controlled current source
Ideal Quantizer	Using Sign function from Verilog-A
Ideal Delay	Using Delay function from Verilog-A
Input Source Clock	Independent voltage source Square wave pulse generator

Table 1. Equivalent Circuit Components used to represent Subblocks

Fig.3 shows the output spectrum from the simulation. The circuit was simulated for 9000 clock cycles and the frequency has been normalized to a sampling rate of 1Hz and input signal with .025 Hz, resulting in an over-sampling ratio (OSR) of 20.

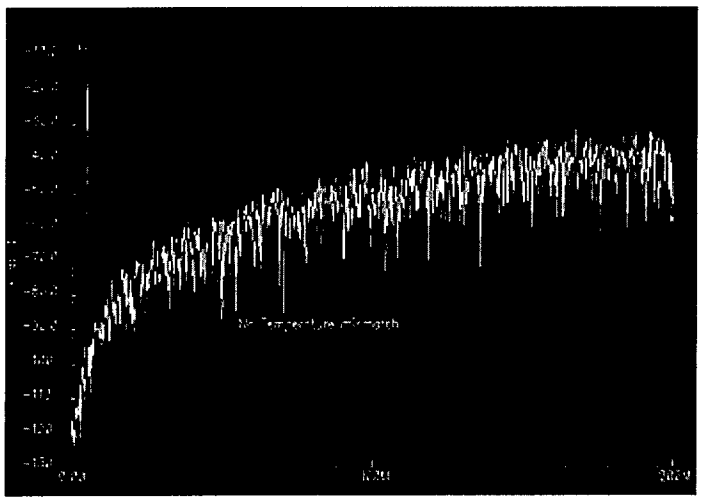


Fig. 3: The output spectrum of 2nd order delta-sigma modulator using a Blackman window

We now list the verilog-A models of the quantizer and delay cell used in our modulator.

```
// VerilogA for NeocadLib, quantizer2, veriloga

`include "constants.h"
`include "discipline.h"

module quantizer(inp,inn,outp,outn,clk);
input inp,inn,clk;
output outp,outn;
electrical inn,inp,outp,outn,clk;

parameter real quantizer_vth=0.0;
parameter real clk_vth=2.5;
parameter real vlevel=0.5;
parameter real trise=1m;
parameter real tfall=1m;

real vout;
real vint;

  analog begin
  @(initial_step) begin
  current_time = 0.0;
  end

  @(cross(V(clk) - clk_vth,1))
   begin
    vint = V(inp,inn);
    current_time = $abstime;
    // 2 input quantizer
    if (vint > quantizer_vth) vout = vlevel;
    else
                        vout = -vlevel;

  end
  V(outp) <+ transition(vout,0,trise,tfall);
  V(outn) <+ transition(-vout,0,trise,tfall);
end
endmodule
```

```
// VerilogA for NeocadLib, delay, veriloga

`include "constants.h"
`include "discipline.h"

module delay_op(out,in);
inout out,in;
electrical out,in;
parameter real td = 1e-3;

analog
  V(out) <+ delay(V(in),td);

endmodule
```

The above models are simulated inside Cadence's spectre simulator. We introduce the non-idealities in the next section.

4.2.2 Introducing Non-Idealities

Delta-Sigma modulators have complex dynamics and therefore, most designers try to isolate individual parameters that affect their performance. The most significant performance metric used is the signal to noise ratio (SNR) at the output of the modulator. High performance modulators have SNR over 120 DB. In our work we focused on the more pressing problem of DAC noise in delta-sigma modulators. DAC noise, unlike other noise sources such as loop filter noise or quantizer noise, is not decreased by the loop filter gain. Therefore, reducing DAC noise is a challenging design problem. To address

this problem, we have introduced two new non-idealities in the model described in the previous section. The DAC, which was modeled using voltage controlled current source, is changed to a current steering differential pair of transistors with an ideal current source. The effect of thermally induced inter-symbol interference is modeled with a verilog-A subblock wrapped around the Bipolar junction transistor (BJT) models. The BJTs are described with a Gummel-Poon SPICE model. Fig. 4 shows the 2nd order delta sigma modulator with non-ideal DAC and a thermally induced inter-symbol interference model wrapped around transistors.

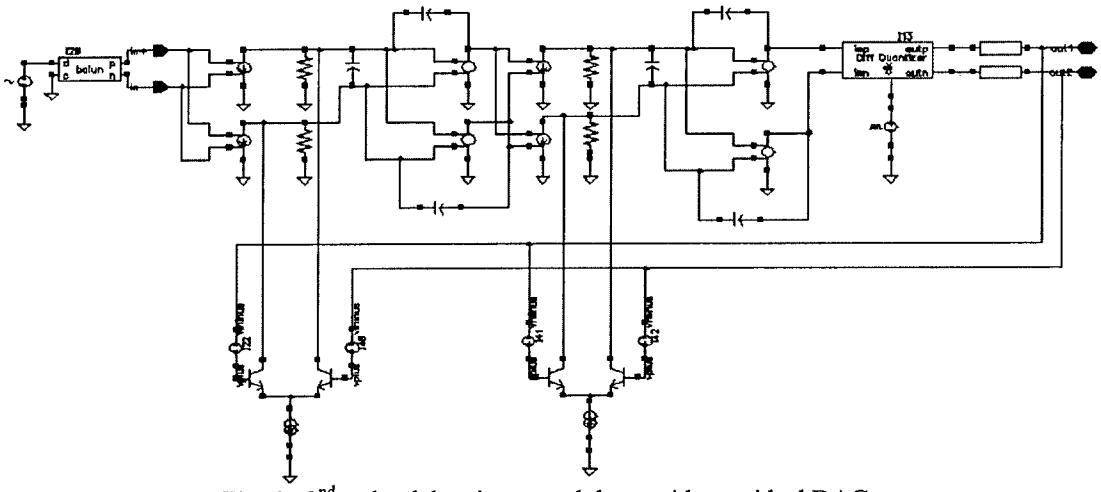


Fig. 4: 2nd order delta-sigma modulator with non-ideal DAC

We now describe thermally induced inter-symbol interference and its effect on the overall performance of the modulator. Thermal-Induced Inter-Symbol Interference is caused by the following phenomenon: two differential signals enter a transistor pair causing the output to switch back and forth as shown in Fig. 6. If the signal into transistor Q_1 switches to high and the signal into Q_2 switches to low, then Q_1 conducts current and Q_2 does not. This causes transistor Q_1 to heat up, and transistor Q_2 to cool down. The next time the signal switches, Q_1 is hotter than Q_2. When a transistor heats up, it needs less of an input voltage to turn on than if it was cool. This means that the input signal to Q_1 must go lower to turn Q_1 **OFF**, and, by contrast, the signal to Q_2 must go higher to turn Q_2 **ON** based on the input signals. Because of this, the new switching time is t later than it would be if the transistors had equal temperature. Δt is assumed to be inversely proportional to the slope of the input signal (i.e. $1/m$) as shown in Fig. 5. If the slope is perfectly vertical, then $t = 0$, regardless of the temperature difference. If the slope is gradual then t would become more sensitive to the transistor temperature differences.

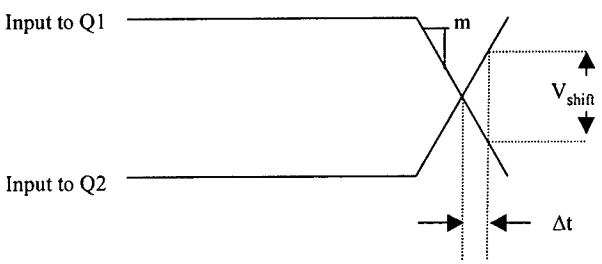


Fig. 5: Differential DAC Input

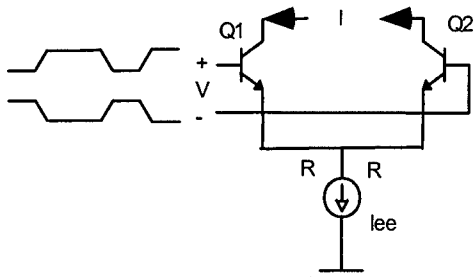


Fig.6: Current Controlled DAC

If T_H is the equilibrium temperature when the transistor in a differential pair is **ON** and T_L is the equilibrium temperature when the transistor is **OFF**, then *T(n)*, the temperature of a transistor Q at the n^{th} clock pulse for the case when it does not change state between consecutive clock cycles (i.e. $s(n-1) = s(n)$) is given by

$$T(n) = T(n-1) + [T_H - T(n-1)](1 - e^{-P/\tau}) \quad \cdots (1)$$

for an **ON** transistor and

$$T(n) = T(n-1) + [T_L - T(n-1)](1 - e^{-P/\tau}) \quad \cdots (2)$$

for an **OFF** transistor. P is the clock period and τ is the thermal time constant. For the case when the transistor makes a transition between consecutive clock cycles (i.e. $s(n-1) = -s(n)$),

$$T(n) = T(n-1) + [T_H - T(n-1)](1 - e^{-P/\tau}) \quad \cdots (3)$$

for a transistor which has just turned **OFF** and

$$T(n) = T(n-1) + [T_L - T(n-1)](1 - e^{-P/\tau}) \quad \cdots (4)$$

for a transistor which has turned **ON**. Using $T_0 = (T_H + T_L)/2$, $T = (T_H - T_L)/2$, we can write simplified equations for the differential pair from Fig. 1:

For the **ON** transistor Q_1

$$T_1(n) = e^{-P/\tau} T_1(n-1) + T_0(1 - e^{-P/\tau}) + \Delta T(1 - e^{-P/\tau}) \quad \cdots (5)$$

and, for **OFF** transistor Q_2

$$T_2(n) = e^{-P/\tau} T_2(n-1) + T_0(1 - e^{-P/\tau}) - \Delta T(1 - e^{-P/\tau}) \quad \cdots (6)$$

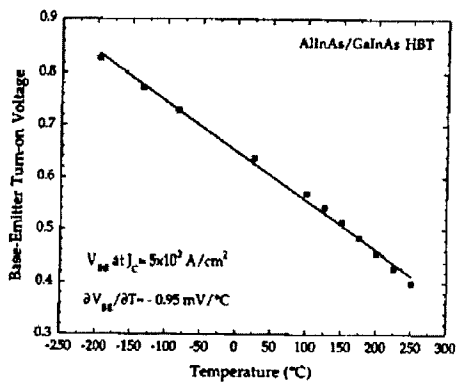


Figure 7: Base Emitter Voltage (V_{be}) vs Temperature

Using the base-emitter voltage (V_{be}) versus temperature curve from Fig. 7 we determine the following relation:

$$c = \frac{\partial V_{be}}{\partial T} \approx \frac{V_{be}(n) - V_{be}(0)}{T(n) - T_{nom}} \quad \cdots \quad (7)$$

c is the rate of change of base-emitter voltage with device temperature ($mV/°C$)

At the point were the DAC switches, $I_1 = I_2 = I_{ee}/2$ and $V_{shift}(n)=V_{be1}(n)-V_{be2}(n)$

$V_{shift}(n)=c(T_1(n)-T_2(n)) \quad \cdots \quad (8)$

$T_1(n)$ and $T_2(n)$ are the device temperature of transistors Q1 and Q2.

Finally from Eq. 6 and fig. 2 we find

$$\Delta t(n) = \frac{V_{shift}(n)}{2m} \quad \cdots \quad (9)$$

To model thermal-induced inter-symbol interference, one needs to change the physical attributes of the transistor (e.g., saturation current) at every clock cycle. Dynamically changing a physical model in a SPICE-based simulator is currently not possible; instead, we model this effect with a perturbation voltage at the base of the transistor. We assume a simplified transistor model with

$$V_{be0} = V_{T0} \ln\left(\frac{I_0}{I_s}\right) \quad \cdots \quad (10)$$

where V_{be0} is the base-emitter voltage and V_{T0} is the thermal voltage at T_{nom}. Therefore from Eq. 7

$$V_{beT}(t) = c(T(t) - T_{nom}) + V_{be0} \cdots (11)$$

The perturbation voltage is added to the DAC as shown in Fig. 4 is $\Delta V = V_{beT} - V_{be0}$

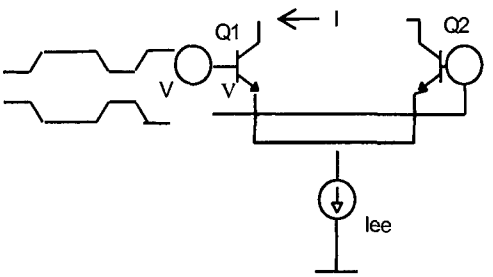


Figure 8: Current Controlled DAC with voltage perturbations

Eq. 11 is modeled using Verilog-A and was simulated in Cadence Spectre simulator. The Verilog-A code is as follows:

```
// VerilogA for NEOTFASTLib, VJT, veriloga
`include "constants.h"
`include "discipline.h"

module VJT(vminus,vplus);
inout vminus,vplus;
electrical vminus,vplus;
```

```
// temperature dependent parameters
parameter real tc = 2e-9; // thermal time constant
parameter real th = 310; // equilibrium temperature (High)
parameter real tl = 300; // equilibrium temperature (Low)
parameter real pw = 125e-12; // pulse width;
parameter real c = 1.11e-3; // thermal coefficient [mV/degC]
real delt,to,tnom;
real real_time,prev_time,step,state;
real prev_devt,curr_devt,dvj;

analog begin
  @ ( initial_step or initial_step("static") ) begin
     to = (th + tl)/2;
     tnom = to;
     delt = (th - tl)/2;
     prev_devt = to;
     curr_devt = prev_devt;
     dvj = 0.0;
   end

  // Modeling the input dependent Device Temperature
   // rising edge of the input
   @ (cross(V(vminus),1.0)) begin
      state = 1;
      step = $abstime - prev_time;
      curr_devt = exp(-step/tc)*prev_devt
                  + to*(1 - exp(-step/tc))
                   - delt * (1 - exp(-step/tc));
      dvj = c*(curr_devt-tnom);
   end
   // falling edge of the input
   @ (cross(V(vminus),-1.0)) begin
      state = -1;
      step = $abstime - prev_time;
      curr_devt = exp(-step/tc)*prev_devt
                  + to*(1 - exp(-step/tc))
                  + delt * (1 - exp(-step/tc));
      dvj = c*(curr_devt-tnom);
   end

   // At every clock pulse
   @ (timer(0,pw)) begin
         step = 0.0;
         prev_time = $abstime;
         prev_devt = curr_devt;
          if (state == 1) begin
            curr_devt = exp(-pw/tc)*prev_devt
                        + to*(1 - exp(-pw/tc))
                        + delt * (1 - exp(-pw/tc));
          end
          else begin
            curr_devt = exp(-pw/tc)*prev_devt
                       + to*(1 - exp(-pw/tc))
                       - delt * (1 - exp(-pw/tc));
          end
   // Changing the emitter base junction potential
```

```
      dvj = c*(curr_devt-tnom);
    end
    V(vminus,vplus) <+ dvj;
  end
endmodule
```

An SHBT device has a thermal resistance of 12.3C/mW. However, this number is complicated by the fact that transistors in a DAC are differential. This means that when one transistor in the DAC heats up, a transistor in close proximity cools down. The substrate is much thicker (about 100 microns) than the distance between the two transistors (about 5 microns) and this means that the thermal resistance for our purposes is actually much less than the above number. This observation leads us to use a thermal time constant that is much less than the published value. In our experiments the thermal time constant (τ) is set at *2 nsec*. Other parameters are as follows:

Table 2: Thermal Induced ISI Model parameters

Quantizer output Rise/Fall Time	50ps/50ps
dVbe,on/dT (c)	1.11 mV/°C
Pulse Width (P)	0.5 sec
Nominal Temperature (T_0)	310°K
Min–Max equilibrium Temp	300°K-320°K

To measure the SNR degradation we sweep variables: T (i.e. $(T_h - T_L)/2$) and rate of change of base-emitter voltage with temperature (i.e. c). The simulator Temperature is made equal to the nominal temperature (i.e. T_0). The plot in Fig.9 shows the input voltage dependent device temperature with T=10°K, Tnom=310°K and τ=2nsec. The plot of the frequency spectrum from the output of the 2nd order CT Σ modulator is shown in Fig.10 for different values of T. The CT Σ modulator has been simulated using Cadence Spectre simulator for 8,000 clock cycles with a single-tone signal of amplitude 200mVat 5MHz. The higher OSR (i.e. $f_s/2f_0$=100) keeps the quantization noise power low and accentuates the effect of noise generated by the thermal mismatch.

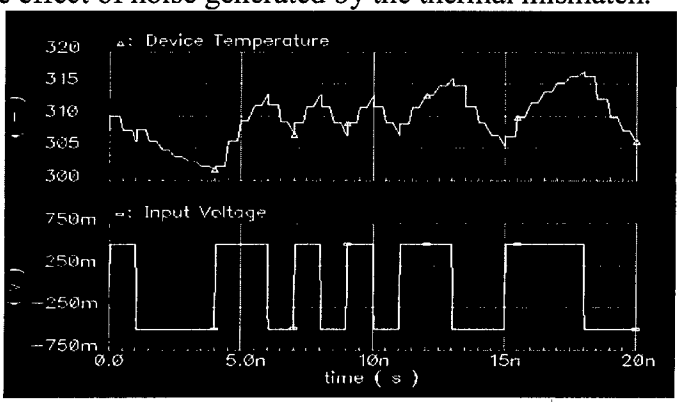


Fig. 9: Signal dependent device temperature change with T=10°K, Tnom=310°K, τ=2nsec

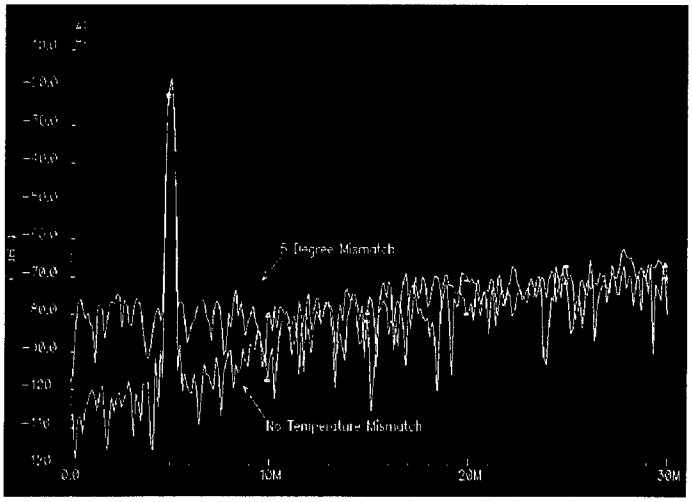


Fig. 10: Frequency Spectrum at T at 0°K, 5°K showing 30db degradation in SNR at OSR=100

Fig. 10 shows the effect of temperature and device mismatches. The devices have been mismatched by 10% by changing β (beta), *Is* (saturation current) and *c* (Rate of turn-on voltage).

The device mismatch parameters are as follows:

Table 3: Device Mismatch Parameters

Device Parameter	Q1	Q2
Saturation Current (Is)	5e-15A	6e-15A
Beta (β)	50	55
dV_{be}/dT	1.11mV/ °C	1.25mV/° C

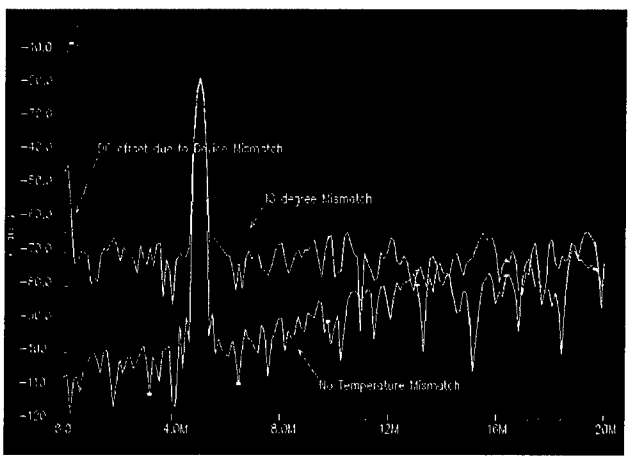


Fig. 11: Frequency spectrum with 10°K Temperature mismatch and 10% Device mismatch

4.2.3 Conclusion

The results from the models show the dependence of SNR in a CT Σ modulator on the thermal characteristics of the transistors. From Fig. 9 one can make the following observations. The SNR degradation in a continuous mode delta-sigma modulator is strongly correlated to the signal dependent temperature mismatch in the DAC. With 5°K temperature mismatch and slope at quantizer output of 20V/ns (i.e. rise/fall time =50ps and Amplitude = +/- 500mV), the maximum Δt shift is 140 femto-seconds. This causes a SNR degradation of approx 30dB as seen from Fig. 10. With 10°K mismatch the SNR degrades further by 20dB as shown in Fig. 11. Any further increase in temperature mismatch can cause the modulator to go unstable.
The top-down methodology introduced in this chapter using verilog-A is a substantial refinement over the older top-down design methodology used at HRL which involved Matlab Simulink and SPICE macro models. Using verilog-A the transition between algorithm and architectural design was more continuous in the design flow and considerable amount of guess work involved in the transition to the transistor level design was reduced.

Chapter 5 Parasitic Extraction Tool

5.1 Introduction

The purpose of the proposed parasitic extraction tool was to estimate in a rigorous manner the network parameterization of the passive interconnect structures of a mixed signal system using a fast frequency-domain fast full-wave electromagnetic simulator. The underlying technique is a high-order method of moment solution with a novel fast iterative solution method, referred to as the Quadrature-Sampled Pre-Corrected FFT (QS-PCFFT) method. In this section, the fundamental integral equation formulation and parameter extraction method is outlined. The QS-PCFFT algorithm is also detailed. Validating results and benchmarking of the tool are also provided.

5.2 Method of Moment Formulation

The interconnect networks of the mixed-signal circuit is assumed to consist of a network of metallic signal and reference lines and ground planes etched within a layered dielectric media. Metal losses are modeled via an impedance boundary condition (non-penetrable conductors), or resistive boundary condition (penetrable conductors). The integral equation formulation used for this physical problem is the mixed potential integral equation (MPIE):

$$\vec{t}\cdot\vec{E}^{inc}(\vec{r}) = Z_s\vec{t}\cdot\vec{J}(\vec{r}) - \vec{t}\cdot\iint_S \bar{\bar{G}}_A(\vec{r},\vec{r}')\cdot\vec{J}(\vec{r}')ds' - \vec{t}\cdot\iint_S \nabla G_V(\vec{r},\vec{r}')\nabla'\cdot\vec{J}(\vec{r}')ds' \quad (0.1)$$

where $\bar{\bar{G}}_A$ is the dyadic vector potential and G_V the scalar potential for a layered media, $\vec{t}$ is a vector tangential to the conductor surface S, and Z_s is the surface impedance of the metallic surface. The dyadic vector potential and the scalar potential are specialized to planar layered media assumed to be infinite in extent [1, 2] and can be expressed in the form of Sommerfeld integrals. In the code developed, the Sommerfeld integrals were pre-computed using a deformed path integration [3], tabulated, and interpolated to controllable accuracy.

The unknown surface current density, $\vec{J}(\vec{r}')$, is calculated using a method of moment solution. The surface is discretized via either curvilinear quadrilaterals or triangles. The unknown surface current density is approximated via high-order divergence conforming basis functions proposed by Graglia, Wilton and Peterson [4]. A Galerkin testing procedure is employed. Consequently the basis and testing functions are identical. This leads to a discrete linear system of equations:

$$\mathbf{Zj} = \mathbf{v} \quad (0.2)$$

where,

$$Z_{m,n} = Z_s\iint_{S_m}\vec{t}_m(\vec{r})\cdot\vec{j}_n(\vec{r})ds - \iint_{S_m}\vec{t}_m(\vec{r})\cdot\iint_{S_n}\bar{\bar{G}}_A(\vec{r},\vec{r}')\cdot\vec{j}_n(\vec{r}')ds' - \iint_{S_m}\vec{t}_m(\vec{r})\cdot\iint_{S_n}\nabla G_V(\vec{r},\vec{r}')\nabla'\cdot\vec{j}_n(\vec{r}')ds', \quad (0.3)$$

$$v_m = \iint_{S_m}\vec{t}_m(\vec{r})\cdot\vec{E}^{inc}(\vec{r})ds$$

j represents the unknown constant coefficient vector weighting the basis functions, S_m is the surface supporting the vector test function $\vec{t}_m$ and S_n is the surface supporting the vector basis function $\vec{j}_n$. Finally, for low frequency stabilization, loop-star basis functions are used [5]. The loop star basis

5.3 Network Parameter Extraction

The method of moment solution presented in the previous section is used to extract the admittance parameters, or Y-parameters, of the mixed signal system network. In order to extract the Y-parameters, one needs to drive a port with an ideal voltage source, while short circuiting all the other loads with an ideal short circuit. Then, by calculating the short circuit currents at all ports, a column of the Y-parameters is calculated. One of the challenges of using a method of moment solution is that we cannot emulate an ideal short circuit, since a finite shorting pin or strip has parasitic inductance and possible radiation loss. Also, one cannot exactly reproduce an ideal voltage source. As a consequence, we need to identify appropriate standards an appropriate de-embedding techniques so that non-physical parasitic affects can be de-embedded from the estimation of the network parameters of the device under test. A novel procedure we have developed to do this is briefly described in this section for *microstrip lines*. Similar procedures can be applied to other types of waveguides, such as co-planar waveguides, slot-lines or striplines.

Consider an N-port network fed by N microstrip signal lines. A port extension of length ℓ is then added to each line beyond the desired reference plane of the port. Each port extension is terminated with an open circuit. A delta-gap source is placed a distance d from the reference plane. Fig. 5.1(a) illustrates the transmission line model for the port extension and an equivalent circuit model. In the equivalent circuit model, Z_s represents the equivalent input impedance of the open circuited stub line beyond the delta-gap source, and $[A_f]$ is the ABCD, or transmission, matrix of the uniform line of length d. Both Z_s and $[A_f]$ are extracted using a method of moment simulation of the actual port extensions, where:

$$[A_f] = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \tag{0.4}$$

Since the port extensions are small, this operation requires nearly negligible computer resources.

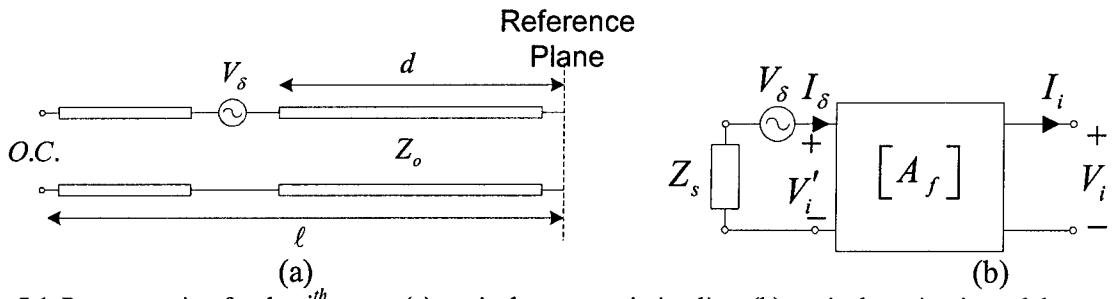


Fig. 5.1 Port extension for the i^{th} port: (a) equivalent transmission line, (b) equivalent circuit model.

Once the port extension parameters are computed, the *Y*-matrix of the N-port network is extracted. As an example, we consider a four-port network. In this simulation, each of the four ports will be excited separately. Consider the excitation of port 1, by a delta-gap voltage source. For the remaining ports, the delta-gap source is short circuited. The method of moment simulation is then used to compute the currents in the reference plane. The voltage at each port cannot be measured directly from the method of moment simulation. Rather, the port voltages are then estimated using the equivalent circuit models in Fig. 5.1(b). To this end, the voltage on the *k*-th port is estimated as:

$$V_k = -I_k^1 Z_k^{ext} \tag{0.5}$$

where, I_k^1 represents the current flowing *into* the *k*-th port due to the delta-gap source voltage on port 1, and Z_k^{ext} represents the effective input impedance of the *k*-th port extension as extracted from the equivalent circuit parameters in Fig. 1 (b). Next, the port voltage on port 1 in the reference plane is computed as:

$$V_1 = F_\delta^1 = \left(V_\delta - Z_s^1 I_\delta^1 - B_1 I_1^1\right) / A_1 \tag{0.6}$$

where Z_s^1 is the input impedance of the open circuit stub for port 1, I_δ^1 is the current at the delta-gap source as computed by the MoM simulation, I_1^1 is the current in the reference plane of port 1 as computed by the MoM simulation, and A_1 and B_1 are ABCD parameters of the feed line in Fig. 1. Given these parameters, the *Y*-matrix is then expressed as:

$$\begin{pmatrix} I_1^1 \\ I_2^1 \\ I_3^1 \\ I_4^1 \end{pmatrix} = \begin{pmatrix} Y_{11} & Y_{12} & Y_{13} & Y_{14} \\ Y_{21} & Y_{22} & Y_{23} & Y_{24} \\ Y_{31} & Y_{32} & Y_{33} & Y_{34} \\ Y_{41} & Y_{42} & Y_{43} & Y_{44} \end{pmatrix} \begin{pmatrix} F_\delta^1 \\ -I_2^1 Z_2^{ext} \\ -I_3^1 Z_3^{ext} \\ -I_4^1 Z_4^{ext} \end{pmatrix} \tag{0.7}$$

Next, a similar calculation is performed for the remaining three ports. Consequently, from this simulation, the *k*-th column of the *Y*-matrix can be computed from the linear system of equations:

$$\begin{pmatrix} I_k^1 \\ I_k^2 \\ I_k^3 \\ I_k^4 \end{pmatrix} = \begin{pmatrix} F_\delta^1 & -I_2^1 Z_2^{ext} & -I_3^1 Z_3^{ext} & -I_4^1 Z_4^{ext} \\ -I_1^2 Z_1^{ext} & F_\delta^2 & -I_3^2 Z_3^{ext} & -I_4^2 Z_4^{ext} \\ -I_1^3 Z_1^{ext} & -I_2^3 Z_2^{ext} & F_\delta^3 & -I_4^3 Z_4^{ext} \\ -I_1^4 Z_1^{ext} & -I_2^4 Z_2^{ext} & -I_3^4 Z_3^{ext} & F_\delta^4 \end{pmatrix} \begin{pmatrix} Y_{k1} \\ Y_{k2} \\ Y_{k3} \\ Y_{k4} \end{pmatrix} \tag{0.8}$$

where I_k^j represents the current in the reference plane of the *j*-th port due to the excitation of port *k* with a delta-gap source, and from (0.6) $F_\delta^k = \left(V_\delta - Z_s^k I_\delta^k - B_k I_k^k\right) / A_k$. Finally, once the *Y*-parameters are computed the scattering parameters can be computed for the *N*-port network directly given the reference impedance for each port.

5.3 Validation of Parasitic Extraction Tool

To validate the parasitic extraction tool, the scattering parameters are extracted from the microstrip low-pass filter in Fig. 5.2. The scattering parameters predicted by the

present method are compared against those predicted using Zeland IE3D™ in Fig. 5.3. Both the magnitude and Phase compare quite well for the two schemes.

A second example is a microstrip meander line, which is a test structure developed by HRL Laboratories. The structure is illustrated in Fig. 5.4 (a) and (b). The meander line circuit has a width of 400 μm and length of about 753 μm. The total length of the meander line is 5055 μm. The microstrip line is 3 μm wide, and is printed under a 700 μm thick InP layer ($\varepsilon_r = 12$, $\tan\delta = .0005$) and on a 1.95 μm thick Polyimid layer ($\varepsilon_r = 4$, $\tan\delta = .001$) backed by a conductor. The scattering parameters were extracted over the frequency range of 0.1 – 75 GHz using the University of Kentucky (UKY) code as well as Zeland IE3D™. The results are compared in Fig. 5.4 (c). There is some deviation in the results. It has been concluded that the deviation is due to the surface impedance approximation. In the UKY code, the broad band surface impedance is based on the formulation presented in [6]. The technique employed by IE3D is unknown.

A third example is that of a 6-pole elliptic microstrip filter with folded hair-pin resonators. The filter is printed on a 32 mil substrate with $\varepsilon_r = 3.38$. Again, the scattering parameters were extracted using the present method and are compared to those computed via Zeland IE3D™. These results are illustrated in Fig. 5.5. Again, quite good agreement is realized.

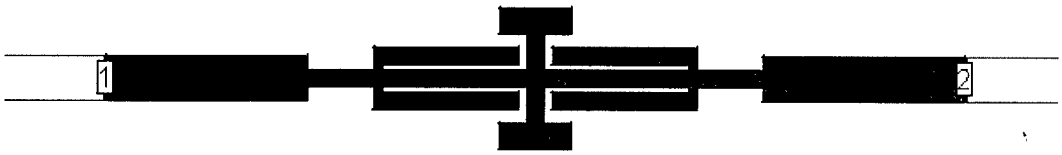


Fig. 5.2 :Top view of a 50 Ω microstrip low-pass filter printed on a 10 mil Alumina substrate ($\varepsilon_r = 9.8$).

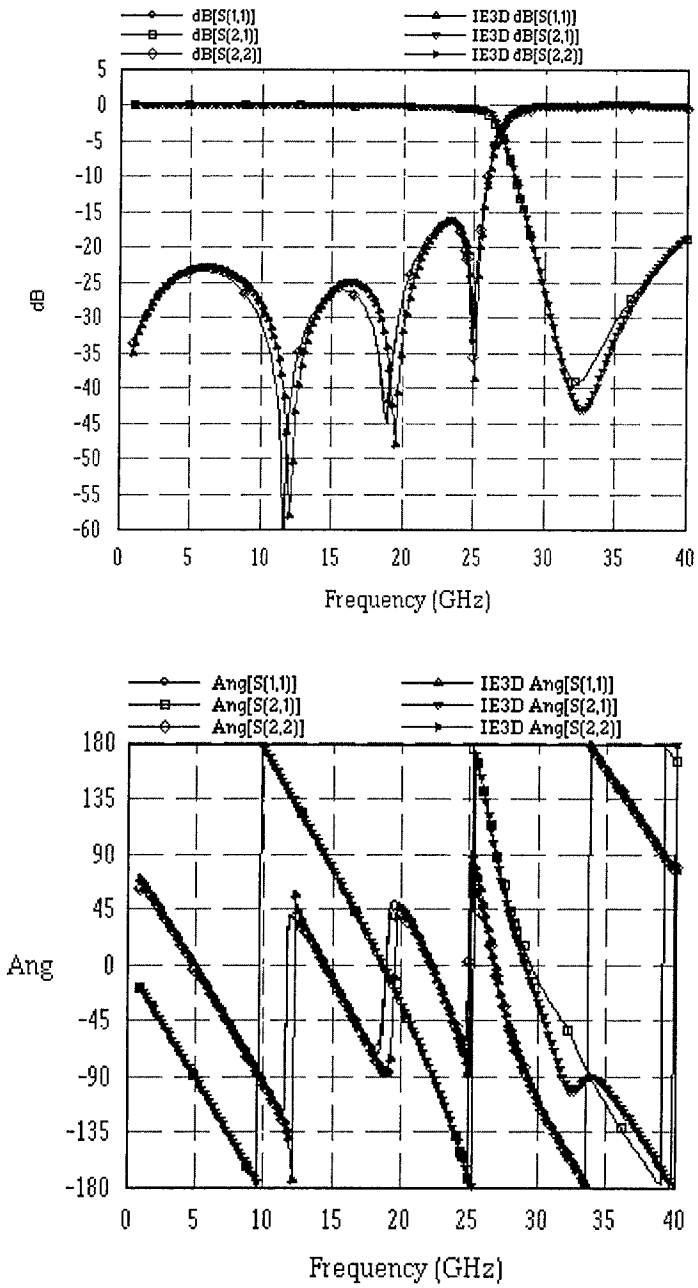
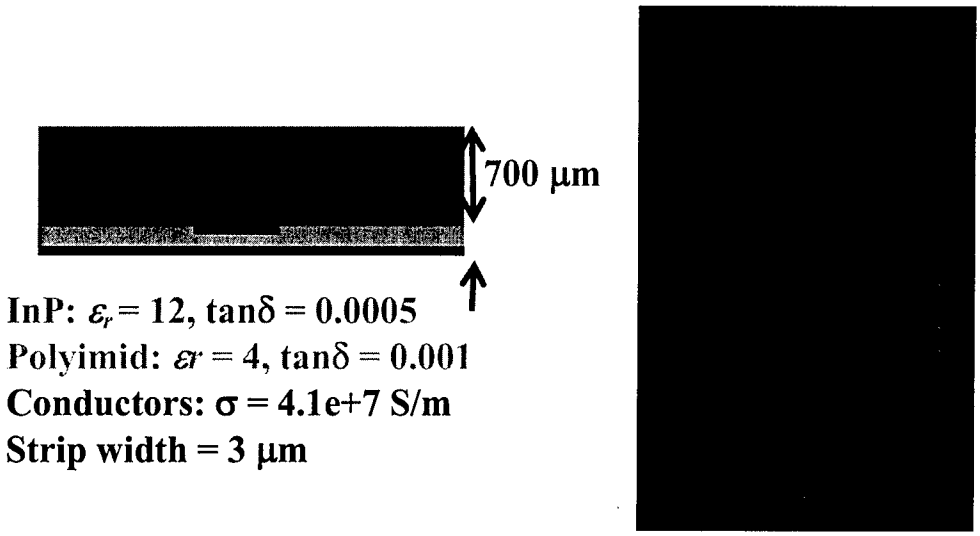
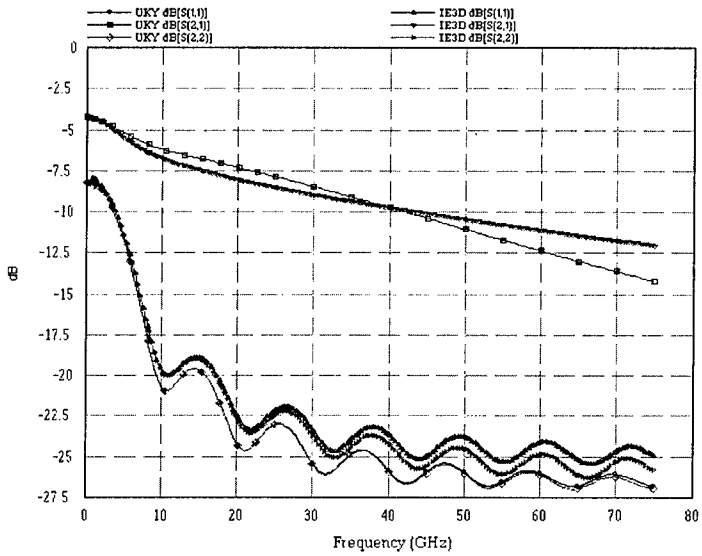


Fig. 5.3 Scattering parameters of the double stub filter compared against IE3D.



(a) (b)



(c)

Fig. 5.4: HRL test structure – the "meander microstrip line." (a) cross section of substrate and microstrip line, (b) top view of the meander line, (c) scattering parameters computed via the UKY simulation code and Zeland IE3D™.

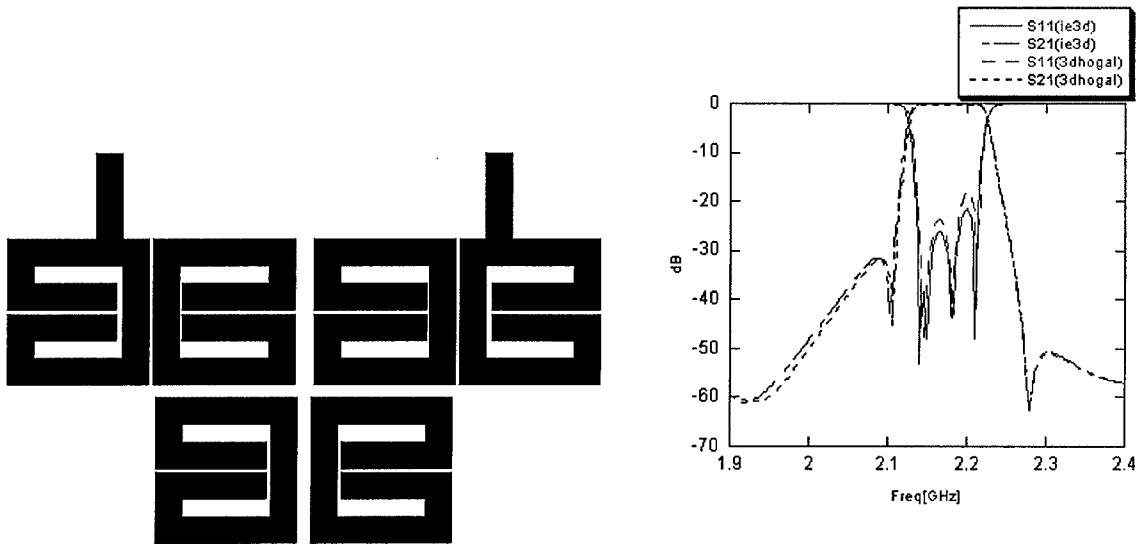


Fig. 5.5: 6 pole elliptic microstrip bandpass filter printed on a 32 mil substrate ($\varepsilon_r = 3.38$) and the scattering parameters computed by the UKY software tool (*3dhogal*) as compared to Zeland IE3D.

5.4 Quadrature Sampled Pre-Corrected FFT Method

The computational resources required to simulate a large structures using a method of moment procedure can be significantly reduced via a fast solution procedure. When employing a high-order scheme, it also becomes imperative to employ a solution procedure that is fast, *and* maintains the high-order nature of the algorithm. In this research program, we have developed a new fast iterative solution method, referred to as the Quadrature Sampled Pre-Corrected FFT (QSPCFFT) [7, 8]. The QSPCFFT method is in the same class of methods as the Adaptive Integral Method (AIM) [9-11] or the Pre-Corrected FFT (PCFFT) method [12]. The specific advantage of the QSPCFFT method is that it does not require the computation of moments to project the currents to the uniform grid. Rather, the discrete Fourier transform (DFT) of the current is evaluated directly using the discontinuous FFT [13] [7], which is based on a quadrature sampling of the currents. The advantage of this scheme is that the DFT of the current is computed to controllable accuracy and is exponentially convergent.

The QSPCFFT is used to accelerate an iterative solution of the method of moment matrix. Each entry of the impedance matrix is expressed as the inner-product of the test vector with the convolution of the MPIE kernel with the basis function. The QSPCFFT is used to accelerate these reactionary terms in the regions where the field cell supporting the test vector and the source cell supporting the basis function vector are sufficiently far removed such that the kernel is sufficiently smooth. In this "far-field" region, the convolution is then computed very efficiently by the inverse Fourier transform of the product of the Fourier transform of the arbitrary current density (computed using the discontinuous FFT [13]) with the Fourier transform of a "windowed" kernel (evaluated using a standard FFT). To describe this in operator form, the impedance matrix is decomposed as:

$$\mathbf{Zj} = \mathbf{Z}^{near}\mathbf{j} + \mathbf{Z}^{far}\mathbf{j} \qquad (0.9)$$

Where $\mathbf{Z}^{near}$ is a sparse matrix that represents interactions of closely separated source/observation cell pairs, and $\mathbf{Z}^{far}$ represents the reaction of souce/observation cell pairs that are sufficiently far removed, and $\bar{j}$ is the unknown vector of the current coefficients. The discontinuous FFT presented in the previous section will be used to evaluate the product $\mathbf{Z}^{far}$. To this end, this product can be computed as:

$$\mathbf{Z}^{far}\mathbf{j} = \mathbf{V}^T H(\mathbf{W}\mathbf{j}) \tag{0.10}$$

where, $\mathbf{W}$ represents a matrix operator that samples the currents with coefficient vector $\mathbf{j}$ at the quadrature points, and maps the vector to $\mathbf{g}$ defined by the uniform FFT grid [7], H is an operator defined as:

$$H(\mathbf{g}) = FFT^{-1}\{FFT(\mathbf{K}) \cdot FFT(\mathbf{g})\} \tag{0.11}$$

where $\mathbf{K}$ is the discrete kernel, and $\mathbf{V}^T$ maps the uniformly sampled field points back to the quadrature points and performs the inner dot product with the test vector using a fixed-point numerical quadrature. Due to the symmetry of the operation, it can be shown that $\mathbf{V} = \mathbf{W}$. Also, from a practical perspective, the discrete kernel $\mathbf{K}$ is windowed such that the numerical singularity near the origin is removed. Furthermore, $FFT(\mathbf{K})$ is performed only once and stored.

The operator in (0.10) is a global operator that computes all source-observation pairs, and hence corrupts the near fields. Thus, the near interaction must be "pre-corrected." This is done as:

$$\tilde{\mathbf{Z}}^{near} = \mathbf{Z}^{near} - \mathbf{W}H\mathbf{V}^T \tag{0.12}$$

It is noted that this can be done efficiently and accurately during the matrix fill analytically via a discrete convolution, or more efficiently using localized FFT's. Finally, the matrix vector product is approximated as:

$$\mathbf{Z}\mathbf{j} = \tilde{\mathbf{Z}}^{near}\mathbf{j} + \mathbf{W}H\mathbf{V}^T\mathbf{j} \tag{0.13}$$

resulting in an algorithm that requires $O(N\log N)$ floating point operations and $O(N)$ storage.

To demonstrate the QSPCFFT algorithm, consider a microstrip patch array antenna, as illustrated in Fig. 5.6. The antennas are printed on a 31 mil substrate with $\varepsilon_r = 4.4$. The scattering parameters of the 2 element array are also illustrated in Fig. 5.6 and are compared to those computed via Zeland IE3D™. The UKY fast solver was then used to simulate the array as the # of elements was increased from 2 to 16. The solution was obtained via the present method and the QSPCFFT algorithm with a BiCGSTAB(l) iterative solver [14] with l = 4. The CPU times required to compute the near field matrix and its pre-corrections are graphed in Fig. 5.7 as a function of N. Also, the CPU time per iteration is also presented. It is noted that on average ~65 iterations per port excitation was required to reach a converged solution. The memory required by the QSPCFFT algorithm is presented in Fig. 5.7 as well and is compared to that required by a full-direct solve (assuming complex double precision for both approaches). Observing these results, it is seen that the fill time scales close to $O(N)$, the cost per iteration scales close to $O(N\log N)$, and the memory as $O(N)$, as expected by the algorithm.

A final example is a spiral inductor array loaded microstrip line, illustrated in Fig. 5.8 (a). The line is printed on a 10 μm InP substrate. Each inductor has a radius of 200 μm. The spacing between inductor load pairs is also 200 μm. The scattering parameters were extracted via the QSPCFFT accelerated method of moment solution over the frequency

range of 0.1 – 100 GHz. These results are illustrated in Fig. 5.8 (c). These results are also compared to those computed via Zeland IE3D. Excellent agreement is observed. Results obtained via the QSPCFFT solution are compared to those obtained via a direct solver in Fig. 5.8 (d). The results compare to better than 2 digits.

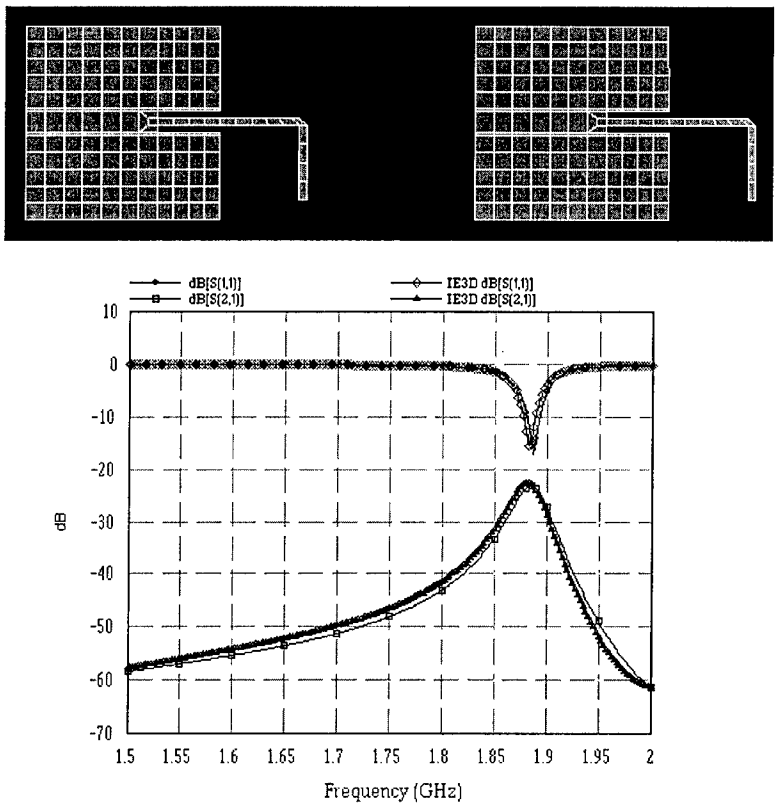


Fig. 5.6 2-element microstrip patch antenna array filter printed on a 31 mil substrate ($\varepsilon_r = 4.4$) and the scattering parameters computed by UKY fast solver and Zeland IE3D.

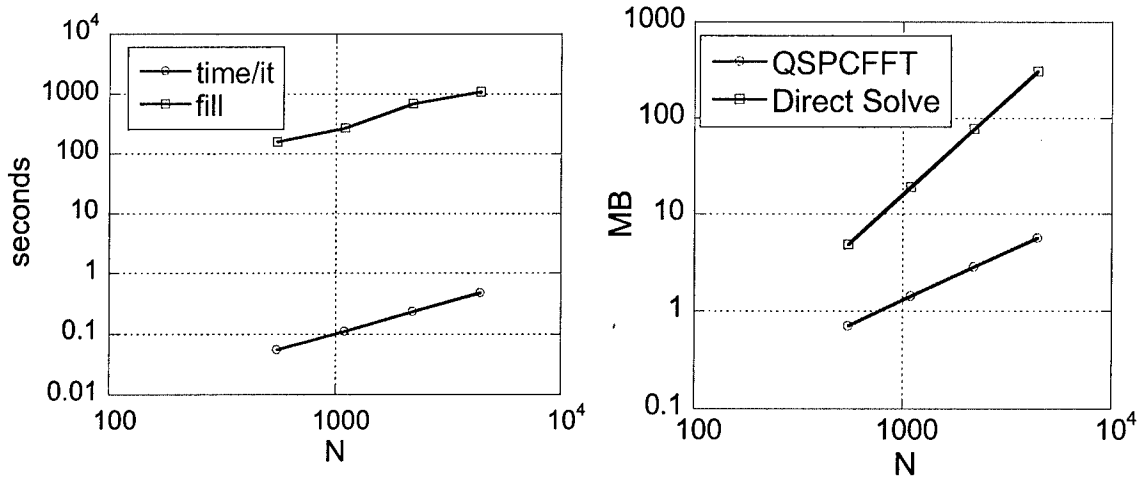


Fig. 5.7 Scaling of CPU time and memory as the patch array in Fig 2 is increased from 2 to 16 patch antennas. The memory compares the QSPCFFT solution with a direct solution method with full-matrix storage.

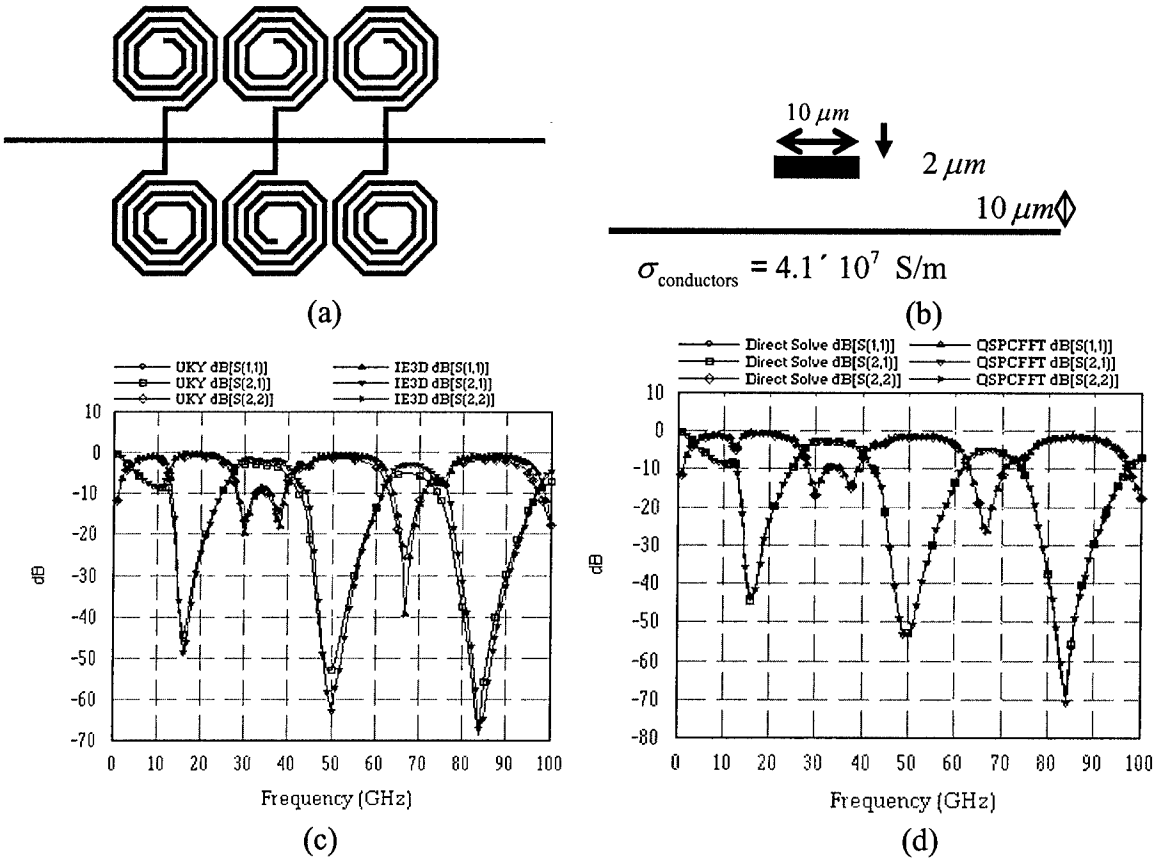


Fig. 5.8 Spiral inductor loaded microstrip line. (a) top view, (b) cross section of the microstrip line and substrate, (c) comparison of the UKY QSPCFFT solution and Zeland IE3D, and (d) comparison of QSPCFFT and direct LU factorization solutions via the UKY software.

[1] K. A. Michalski and D. L. Zheng, "Electromagnetic Scattering and Radiation by Surfaces of Arbitrary Shape in Layered Media.1. Theory," *IEEE Transactions on Antennas and Propagation*, vol. 38, no. 3, pp. 335-344, Mar 1990.

[2] K. A. Michalski and J. R. Mosig, "Multilayered media Green's functions in integral equation formulations," *IEEE Transactions on Antennas and Propagation*, vol. 45, no. 3, pp. 508-519, Mar 1997.

[3] P. Gay-Balmaz and J. R. Mosig, "Three-dimensional planar radiating structures in stratified media," *International Journal on Microwave and Millimeter-Wave Computer Aided Engineering*, vol. 7, no. September, pp. 330-343, 1997.

[4] R. D. Graglia, D. R. Wilton, and A. F. Peterson, "Higher order interpolatory vector bases for computational electromagnetics," *IEEE Transactions on Antennas and Propagation*, vol. 45, no. March, pp. 329-342, 1997.

[5] J. F. Lee, R. Lee, and R. Burkholder, "Loop star basis functions and a robust preconditioner for EFIE scattering problems," *IEEE Transactions on Antennas and Propagation*, vol. in press, no., 2003.

[6] J. C. Rautio and V. Demir, "Microstrip conductor loss models for electromagnetic analysis," *IEEE Transactions on Microwave Theory and Techniques*, vol. 51, no. 3, pp. 915-921, Mar 2003.

[7] S. D. Gedney, A. Zhu, W.-H. Tang, G. Liu, and P. Petre, "A Fast, High-Order Quadrature Sampled Pre-Corrected FFT for Electromagnetic Scattering," *Microwave and Optical Technology Letters*, vol. 36, no. 5, pp. 343-349, March 5 2003.

[8] A. Zhu and S. D. Gedney, "A Quadrature Sampled Pre-Corrected FFT Method for the Electromagnetic Scattering from Inhomogeneous Objects," *IEEE Antennas and Wireless Propagation Letters*, vol. 2, no. 1, pp. 50-53, 2003.

[9] E. Bleszynski, M. Bleszynski, and T. Jaroszewicz, "AIM: Adaptive Integral Method for Solving Large-Scale Electromagnetic Scattering and Radiation Problems," *Radio Science*, vol. 31, no. 5, pp. 1225-1251, 1996.

[10] S. S. Bindiganavale, J. L. Volakis, and H. Anastassiu, "Scattering from planar structures containing small features using the adaptive integral method (AIM)," *IEEE Transactions on Antennas and Propagation*, vol. 46, no. 12, pp. 1867-1878, Dec 1998.

[11] H. T. Anastassiu, M. Smelyanskiy, S. Bindiganavale, and J. L. Volakis, "Scattering from relatively flat surfaces using the adaptive integral method," *Radio Science*, vol. 33, no. 1, pp. 7-16, Jan-Feb 1998.

[12] J. R. Phillips and J. K. White, "A precorrected-FFT method for electrostatic analysis of complicated 3-D structures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 10, pp. 1059-1072, 1997.

[13] G. X. Fan and Q. H. Liu, "The CGFFT method with a discontinuous FFT algorithm," *Microwave and Optical Technology Letters*, vol. 29, no. 1, pp. 47-49, 2001.

[14] G. L. G. Sleijpen and D. R. Fokkema, "BICGSTAB(*l*) for linear equations involving matrices with complex spectrum," *Electronic Transactions on Numerical Analysis*, vol. 1, no. 1, pp. 11-32, 1993.

CHAPTER 6 IMPACT ON MIXED SIGNAL IC DESIGN FLOW AND VALIDATION

6.1 Impact And Validation

Despite the huge investment in CAD software infrastructure, analog design is still largely a manual and oftentimes painful process that gets exponentially harder as the circuit becomes more complex. NeoCAD was designed to directly impact this problem by enabling a faster process flow that leads to fewer errors at tape-out.

To understand the impact of the NeoCAD effort, we first look at a standard analog IC design flow in Fig. 6.1. The design progresses from a system architecture to a functional design, to a transistor-level circuit design. The circuit is then laid out, and based on the parasitics of the layout, the circuit is once again modified. The process is quite lengthy and entails many compromises. For example, if a fixed amount of time and resources are available to design the circuit, the circuit may not be fully tested before release; alternately, if the circuit is carefully tested, the design may take too long to complete. The major bottleneck in testing and verification is the simulation times. By reducing the simulation times, we can make more tests in a given time and hence improve our chances of first-pass success. NeoCAD addressed fundamental issues in each of these blocks to improve speed and accuracy. While preceding sections cover the results of the NeoCAD project in more detail, this section provides selected examples to illustrate its impact on the design flow.

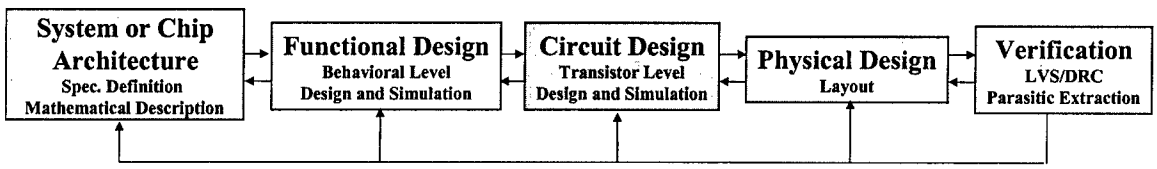


Fig. 6.1: Analog circuit design flow.

6.1.1 Functional Design Improvements

Fig. 6.2 shows the design flow before and after NeoCAD. Before the NeoCAD project, we created a circuit, such as a $\Delta\Sigma$ modulator, first at the ideal behavioral-level, and after verification, moved directly to a transistor-level model that we simulated in Cadence. This jump led to many problems. First, verification at the transistor-level was very lengthy because the circuits tend to be very complex and take a long time to simulate. Because the verification is so lengthy, it became time-prohibitive to fully simulate, debug and optimize the transistor-level circuit. As a result, there is a reasonable likelihood problems will not get caught before the design is released to fabrication.

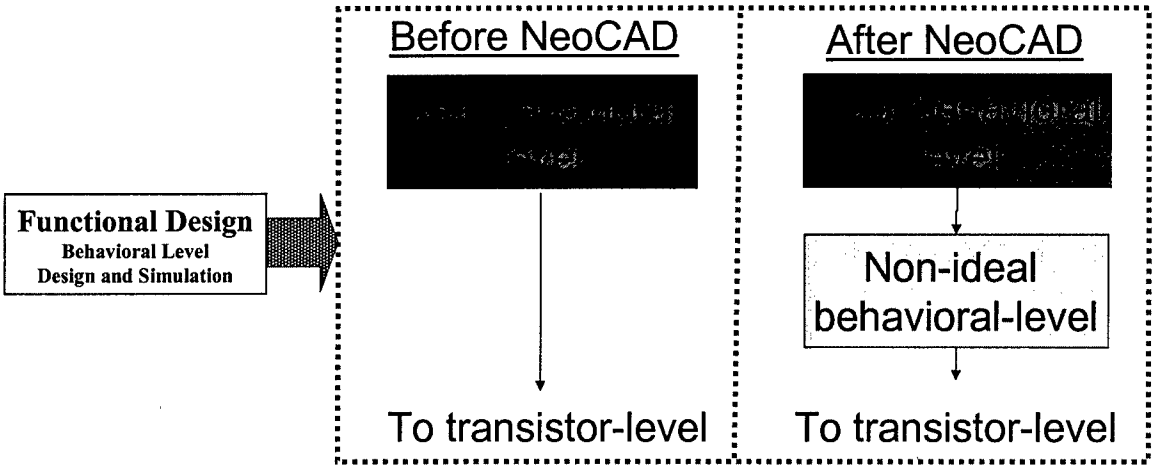


Fig. 6.2: Flow of functional design.

NeoCAD addressed this weakness by adding another design step, which we call the non-ideal behavioral-level step. In this model, we can account for a large amount of the transistor-level behavior without resorting to lengthy transistor-level simulations. The result is that many features of the circuit are optimized at this level and as a result far less time needs to be spent with transistor-level optimization.

Fig. 6.3 shows a 4^{th}-order bandpass $\Delta\Sigma$ modulator, where we show the nonlinear behavioral models we have developed for this project.

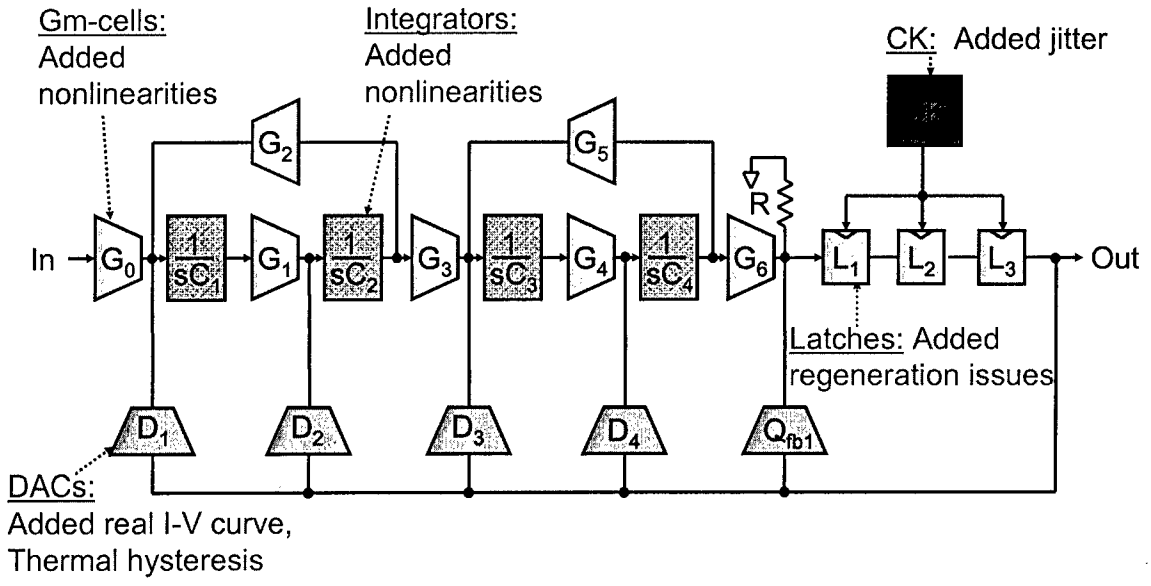


Fig. 6.3: $\Delta\Sigma$ modulator with nonideal behavioral-level models.

The nonlinear behavioral-level models have allowed us to run a large amount of simulations in a reasonable time. One example of how these models have improved our circuit is in the optimization of Q_{fb}. The original value of Q_{fb} is mathematically derived

to optimize performance from an idealized ΔΣ modulator. This model, however, does not include effects such as quantizer metastability that result from the latch RC time constant. As the joint effect of the RC time constant and Q_{fb} on the SNR is currently not solvable analytically, we would nominally run a series of simulations to see the effect of this nonideality on the SNR. However, the simulation times are prohibitive to do such an analysis at the transistor level.

We are, however, able to perform the simulations using the non-ideal behavioral-level models developed under NeoCAD. It shows that the original value of Qfb does, in fact, lead to optimal SNR values. However, small positive deviations in Qfb lead to a large degradation in the SNR, and oftentimes, causes the circuit to go unstable. By simply reducing the value of Qfb, the SNR drops slightly but the circuit becomes much more robust. Thus, as a result of the nonlinear behavioral-level models, we were able to make our circuit more robust than would be possible using the conventional design methodology.

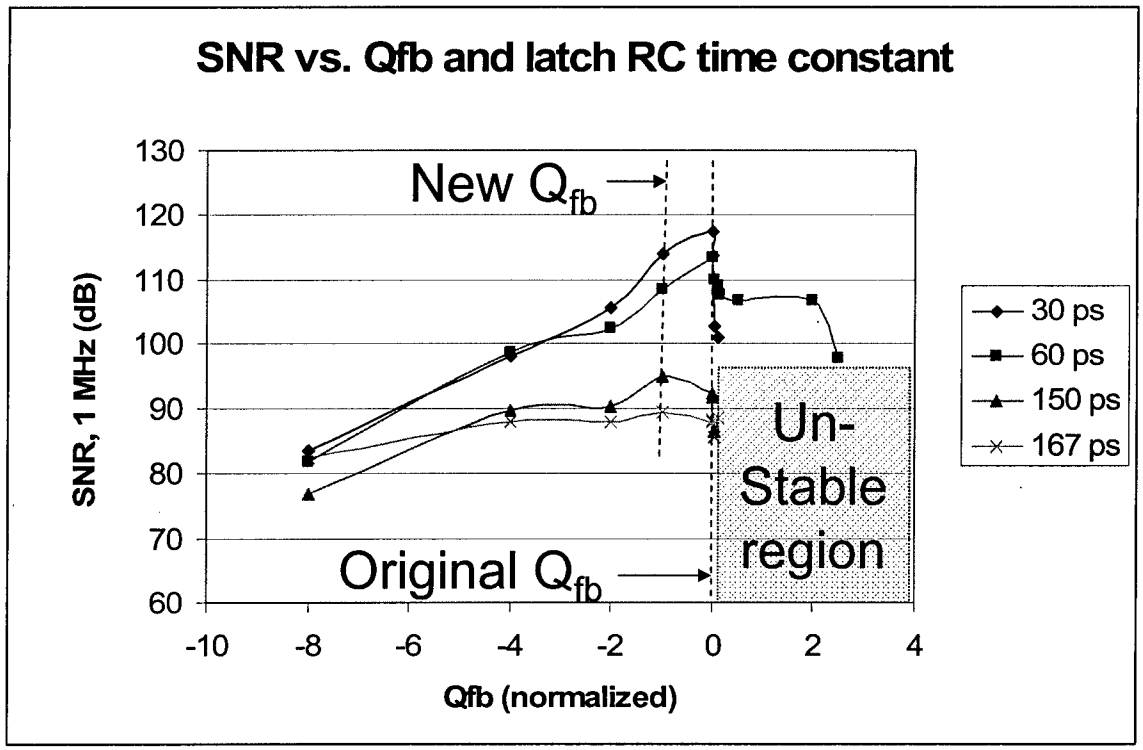


Fig. 6.4: SNR of ΔΣ modulator as function of Q_{fb} and latch RC time constant.

It is important to recognize the general utility of these models. There are currently very few bandpass ΔΣ modulators being designed in industry. However, these ΔΣ modulators are designed using basic building blocks that are very common throughout the industry. Because of this, effort developing the nonlinear models can pay off in a variety of other circuits. Fig. 6.5, for example, shows a tunable DAC mismatch shaper. This circuit is used to improve the SNR of a DAC by shifting the mismatch error spectrum out-of-band. While performing a largely different function, there are many common blocks that have nonlinear models developed on the NeoCAD project.

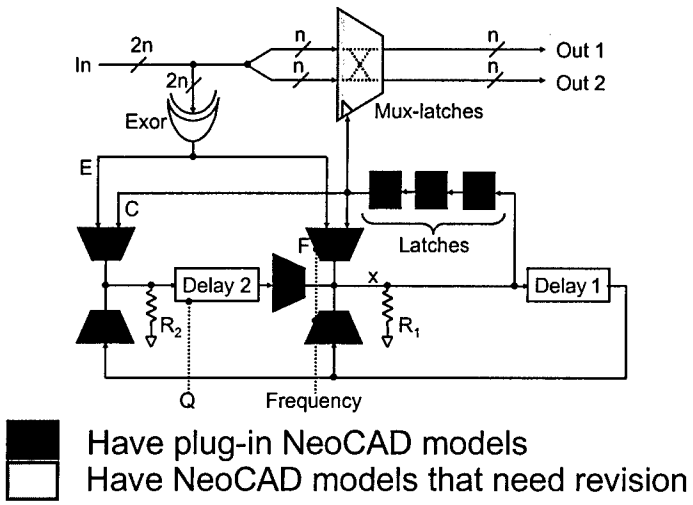


Fig. 6.5: DAC mismatch shaper circuit.

6.1.2 Simulation Time Improvements

A major effort of the NeoCAD project is to reduce the simulation time for a given set of models. HAARSPICE could enable 10X faster simulations under a given set of test conditions. This speed improvement manifests itself in two ways. First, the design cycle time can decrease. Second, for a given design time allowed, the HAARSPICE speed improvement can lead to more design iterations and a more robust circuit.

We see an example of this in Fig. 6.6, which shows the output of a 2-tone test. In Fig. 6.6 (a), time constraints limit the run length of the time-domain simulation, and the FFT yields little information. The two tones are not yet clearly distinguished. By contrast, an 8X longer simulation time can yield an FFT shown in Fig. 6.6 (b). The input tones and third-order harmonics are clearly visible. Because of the accurate determination of the nonlinearity, it is easier to debug and optimize the circuit.

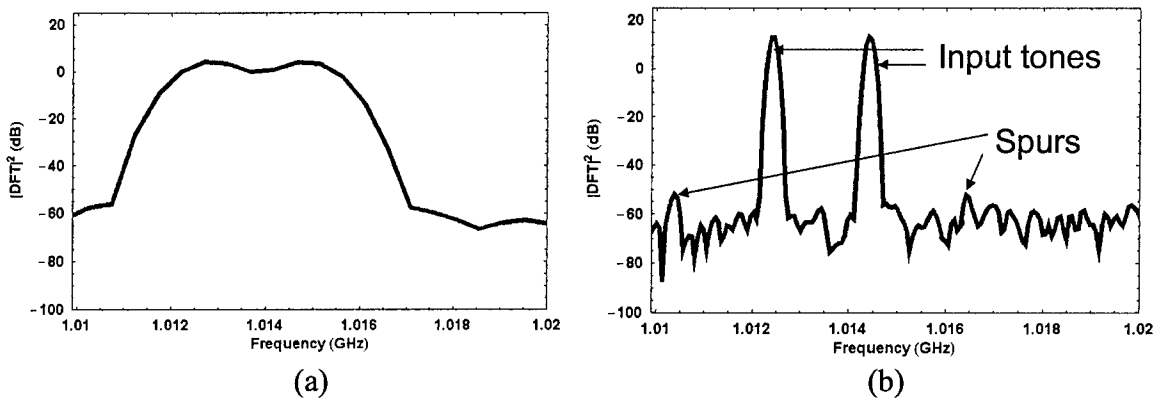


Fig. 6.6: FFT of time-domain output for (a) 8K clock cycles, and (b) 64K clock cycles. The simulation yields far more information with more FFT points.

6.1.3 Parasitic Extraction Improvements

After transistor-level design and layout, simulations are performed to see the effect of parasitic resistances, inductances, and capacitances on the performance of the circuit. Fig. 6.7 shows the flow before and after BEMP. Before BEMP, parasitic extraction was performed with Cadence or Sonnet. Both of these programs had severe shortcomings. Cadence has a piecewise model that models each wire section as a circuit element. If there are many sections to the wire, the resultant model becomes extremely complicated. Secondly, long sections are treated with the same precision as short sections, and hence the models of the long sections are less precise than the short sections, even though the long sections need to be more precise. Thirdly, an LRC extraction causes so many nodes that realistic post-layout simulations cannot be performed on complex circuits. Finally, Cadence's extractor functions in such a way that only nearest-neighbor interactions can be modeled. Sonnet's EM-based software avoids these drawbacks but has an overly simplified model that uses only one inductor, capacitor and resistor between a line's input and output. The model is hence accurate at only one frequency. BEMP, by contrast, uses an EM-based distributed model that is accurate across a wide bandwidth. Importantly, this increase in accuracy does not seem to cost a significant amount of simulation time. In fact, the BEMP-based model could offer up to a 10X reduction in simulation time over those performed using Cadence's LRC-based parasitics.

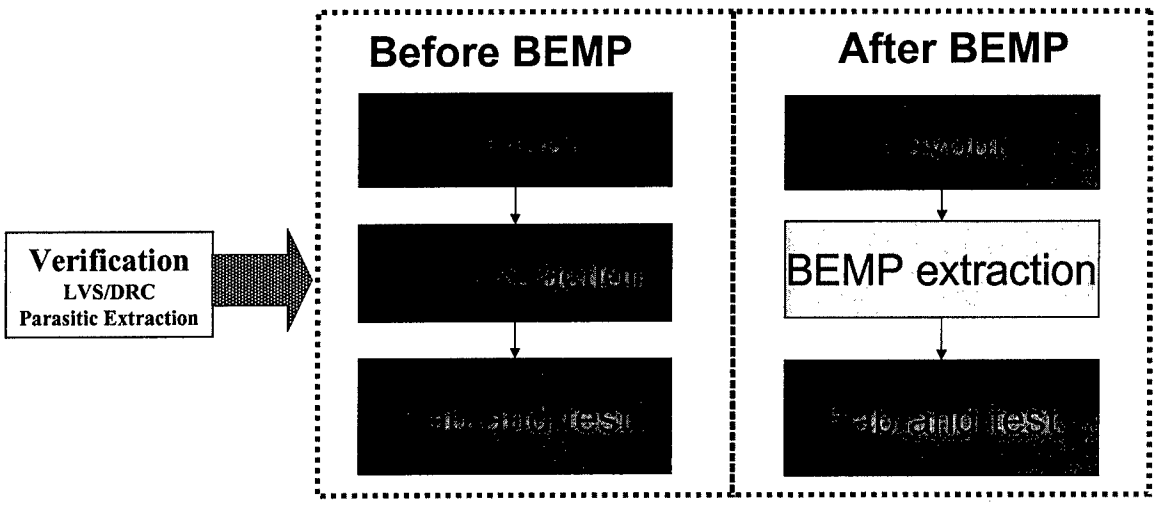


Fig. 6.7: Verification process before and after BEMP.

To verify the accuracy of BEMP extraction vs. Sonnet extraction, we used a 32-bit bus line on an ADC, shown in Fig. 6.8.

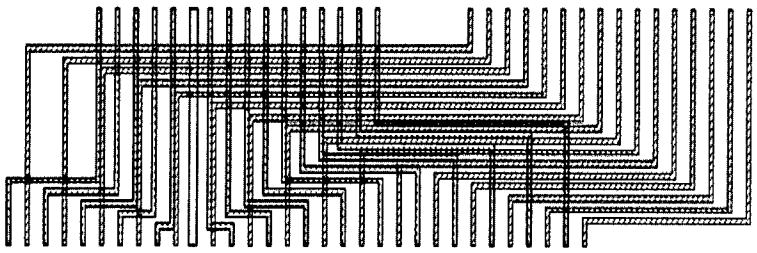


Fig. 6.8: Bus line on ADC. Width of structure is about 800 μm.

A tapered square wave was then fed into this bus to test for ringing. The results are shown in Fig. 6.9. The BEMP response is somewhat obscured because the peaks lie on the lines 1.0 and -1.0, but overall show far less ringing with the Sonnet model. This then raises the question of which model is more accurate. Further runs with Sonnet demonstrated that the ringing was highly dependent on the frequency from which the parasitics were extracted. This is a natural problem which results from an extremely simple one-stage LRC model used by Sonnet. The Sonnet model, extracted at high frequencies, generated very little ringing. Since the voltage transitions have very high-frequency components, this supports the results from BEMP model that show the line has little true ringing. Because of this result, the bus was determined to be satisfactory and an otherwise lengthy optimization process was avoided.

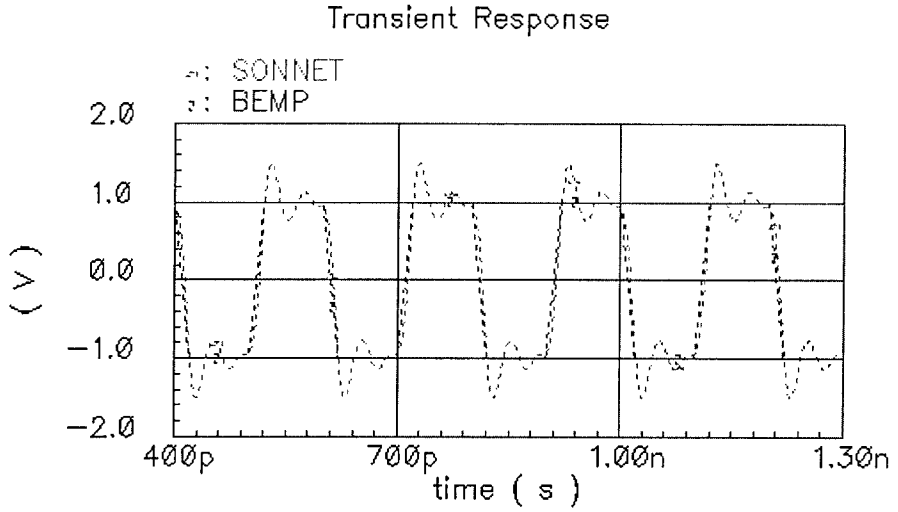


Fig. 6.9: Output of square pulse on bus using Sonnet and BEMP models.

6.2 Conclusions

NeoCAD led to several key improvements in the analog design flow. By implementing non-ideal behavioral-level models, NeoCAD allowed for a good deal of optimization to be performed early in the process. Using HAARSPICE, simulations could run up to 10X faster, enabling more simulations in a fixed design time. Finally, BEMP allowed faster, more accurate parasitic extraction than was available using Cadence or Sonnet. As a result of these efforts, NeoCAD enabled HRL to design more robust circuits faster.

CHAPTER 7 APPENDIX

7.1 HAARSPICE User's Manual

7.2 BEMP User's Manual

7.3 Published Papers and Presentations

HAARSPICE

Introduction

HAARSPICE circuit simulator is a suite of programs used for Time domain and DC analyses of linear and nonlinear analog circuits and systems. **HAARSPICE** was developed at HRL LLC (Malibu, CA) under the DARPA-NEOCAD contract. \

HAARSPICE suite of programs, run on Linux 7.02 and have a text-based interface. **HAARSPICE** accepts ascii text netlist of electronic circuits generated using Cadence's Schematic capture environment.

HAARSPICE *Package Programs*

Typing in "**haarSPICE**" on the command line prompt, the following options are displayed

```
enl - Eigenspace non-linear solver (jacobian)
eld - Eigenspace linear domain solver
ese - Eigenspace evolution nl-domain solver(jacobian free)
dcs - DC sweep
snlc- Supported netlist components
dcp - DC Point of circuit at t = 0
```

In the next sections we shall go over each of them briefly and with the aid of an example highlight their attributes. The example used is a 4^{th} order continuous time band-pass delta-sigma modulator designed at HRL.

Components

Being a Research code **HAARSPICE** is often updated to support more components. The current status of **HAARSPICE** supported components can be checked by typing **snlc** (i.e. supported netlist components), on the command line prompt.

available netlist components:

```
tag (...) atod|AtoD parameters ???
tag (...) bjt|NPN|PNP parameters ???
tag (nodeA nodeB) capacitor c=value "value of capacitance between
nodeA and nodeB"
tag (...) cccs|cccs_hdl parameters ???
tag (...) ccvs parameters ???
tag (...) delay|delay_op parameters ???
tag (...) quantizer3|differentialQuantizer parameters ???
tag (...) diode|diode_hdl|diode_sch parameters ???
tag (...) dsmod|delta-sigma-modulator parameters ???
tag (...) dtoa|DtoA parameters ???
tag (nodeA nodeB) inductor l=value "value of inductance between
nodeA and nodeB"
tag (...) isource|csource parameters ???
tag (...) quantizer1|quantizer parameters ???
tag (nodeA nodeB) resistor r=value "value of resistance between
nodeA and nodeB"
tag (...) sign parameters ???
tag (nodeA1 nodeA2 nodeB1 nodeB2) transformer n1=value n2=value2
"value1/value2 is gain between nodes A and nodes B"
tag (...) vccs parameters ???
tag (...) vcvs parameters ???
tag (...) vsource parameters ???
```

??? denotes instance parameters

Using the listed components, the user can generate the netlist of circuits. In the example shown we have the netlist generated with Cadence's schematic capture software. Currently we support linear and non-linear components. The devices supported are PN Junction Diode, NPN and PNP-type Bipolar Junction Transistor (BJT)

The models used are large-signal nonlinear and can simulate DC and transient domain circuit responses. For BJT we use a Large Signal Gummel-Poon type

model and for the diode a charge based model has been used. The next section gives detail description of each component and its attributes.

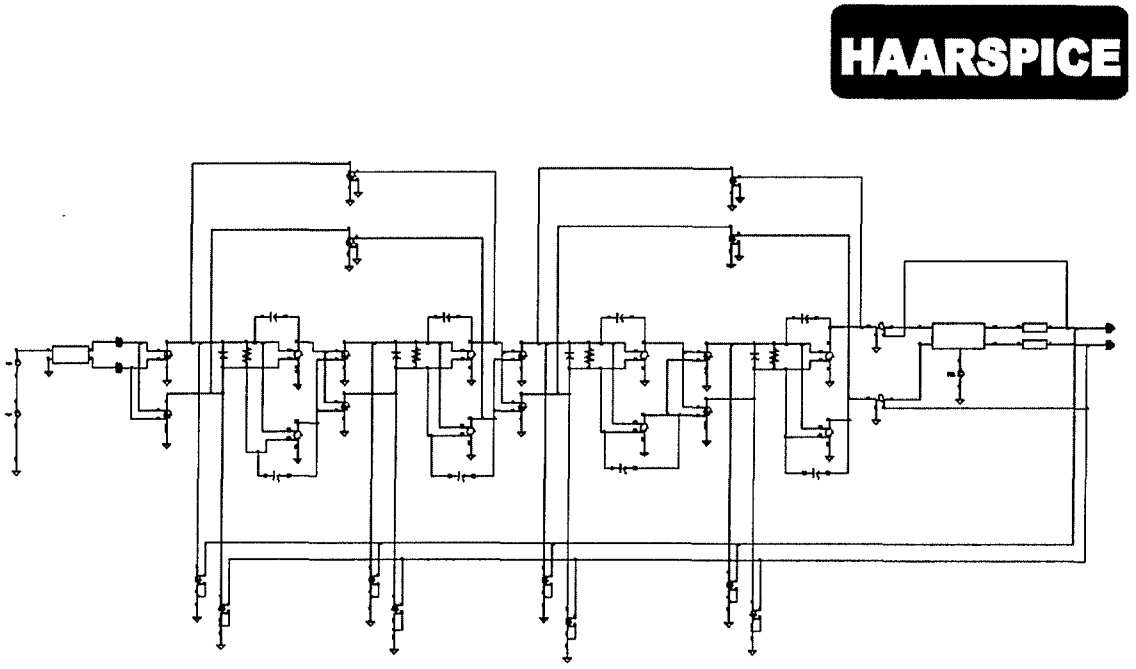


Fig. 1: Schematic of 4th order delta-sigma modulator with linear components

Netlist Generated using Cadence's Schematic Capture Tool

```
parameters Qfb=-0.5*164.6m K4=-0.5*82.55u K3=0.5*3.332m K2=0.5*1.933m \
    K1=0.5*4.833u gmb2=-1.0*6.383m gmb1=-1.0*6.383m gm4=0.5*6.383m \
    gm3=0.5*2.5m gm2=0.5*6.383m gm1=0.5*1.66m C4=1p C3=1p C2=1p C1=1p \
    A4=-100 A3=-100 A2=-100 A1=-100
parameters mgm1=-gm1
parameters mgm2=-gm2
parameters mgm3=-gm3
parameters mgm4=-gm4
parameters mA1=-A1
parameters mA2=-A2
parameters mA3=-A3
parameters mA4=-A4
// Library name: ee101Lib
// Cell name: balun
// View name: schematic
subckt balun c d n p
    _inst0 (d 0 p c) transformer n1=2
    K0 (d 0 c n) transformer n1=2
ends balun
// End of subcircuit definition.

// Library name: NeocadLib
// Cell name: SD4D
// View name: schematic
I39 (0 net0291 _net2 _net1) balun
```

```
I13 (net156 net124 net8 net7 net168) quantizer3 quantizer_vth=0.0 \
        clk_vth=0
G7 (net44 0 net104 net112) vccs gm=mgm4
G6 (net56 0 _net3 0) vccs gm=K3
Gmb2 (net56 0 net46 0) vccs gm=gmb2
GMB1d (net60 0 net50 0) vccs gm=gmb1
```


```
G4 (net64 0 _net3 0) vccs gm=K2
G3 (net56 0 net82 net50) vccs gm=mgm3
G8 (net44 0 _net3 0) vccs gm=K4
GM1d (net60 0 _net1 _net2) vccs gm=mgm1
GM2d (net64 0 net98 net94) vccs gm=mgm2
DAC1d (net60 0 _net3 0) vccs gm=K1
GM1 (net68 0 _net1 _net2) vccs gm=gm1
DAC1 (net68 0 _net0 0) vccs gm=K1
DAC2 (net72 0 _net0 0) vccs gm=K2
GM2 (net72 0 net98 net94) vccs gm=gm2
DAC3 (net76 0 _net0 0) vccs gm=K3
GM3 (net76 0 net82 net50) vccs gm=gm3
GM4 (net80 0 net104 net112) vccs gm=gm4
DAC4 (net80 0 _net0 0) vccs gm=K4
Gmb1 (net68 0 net82 0) vccs gm=gmb1
G12 (net76 0 net86 0) vccs gm=gmb2
C7 (net44 net46) capacitor c=C4
C2d (net64 net50) capacitor c=C2
C1d (net60 net94) capacitor c=C1
Ci1 (net68 net60) capacitor c=100.0f
_inst4 (net68 net98) capacitor c=C1
_inst5 (net72 net82) capacitor c=C2
Ci2 (net72 net64) capacitor c=100.0f
C8 (net76 net104) capacitor c=C3
Ci3 (net76 net56) capacitor c=100.0f
Ci4 (net80 net44) capacitor c=100.0f
_inst6 (net80 net86) capacitor c=C4
C6 (net56 net112) capacitor c=C3
Ri1 (net68 net60) resistor r=1M
Ri2 (net72 net64) resistor r=1M
Ri3 (net76 net56) resistor r=1M
Ri4 (net80 net44) resistor r=1M
E3 (net46 0 net80 net44) vcvs gain=mA4
Av2d (net50 0 net72 net64) vcvs gain=mA2
E2 (net112 0 net76 net56) vcvs gain=mA3
E4 (net124 net46 _net3 0) vcvs gain=Qfb
Av1 (net98 0 net68 net60) vcvs gain=A1
Av2 (net82 0 net72 net64) vcvs gain=A2
Av3 (net104 0 net76 net56) vcvs gain=A3
Av4 (net86 0 net80 net44) vcvs gain=A4
E7 (net156 net86 _net0 0) vcvs gain=Qfb
Av1d (net94 0 net68 net60) vcvs gain=mA1
V0 (net0291 0) vsource type=sine ampl=250.0m freq=998M
```

```
I34 (_net3 net7) delay_op td=125p
I19 (_net0 net8) delay_op td=125p
V1 (net168 0) vsource type=pulse val0=-1 val1=1 period=250p rise=0p \
        fall=0p width=125p
linearDomain start=0 stop=125p
.end
```

The above netlist is a representation of the 4th order delta-sigma modulator with linear components. The 1-bit quantizer used is the only non-linearity in the circuit. We next explain the syntax rules of some of the components in the netlist.

Linear Components

Resistor

Syntax: *Name N+ N- resistor parameter=value*

N+, N-: nodes
Supported parameter*: r*
value : Resistance (Ω)

Maximum resistance supported is $1/G_{MIN}$

Capacitor

Syntax: *Name N+ N- capacitor parameter=value*

N+, N-: nodes
Supported parameter*: c*
value : Capacitance (Farads)

Inductor

Syntax: *Name N+ N- inductor parameter=value*

N+, N-: nodes
Supported parameter*: l*
value : Inductance (Henry)

Ideal Transformer
Syntax: *Name NP+ NP- NS+ NS- xfmr n1=value n2=value*

NP+, NP-: primary nodes
NS+, Ns-: secondary nodes
Supported parameters:
n1: Number of turns in primary winding
n2: Number of turns in secondary winding

An ideal transformer is modeled, so it acts as a transformer at DC

Independent Voltage Source

Syntax: *Name N+ N- vsource parameter=value*

N+, N-: nodes

Supported parameters:
type= dc, pulse, sine
freq=Frequency of sinusoidal waveform (Hz)
ampl=Peak amplitude of sinusoidal waveform (Volts)
rise=Rise time for pulse waveform (seconds)
fall=Fall time for pulse waveform (seconds)
period= Period of waveform (seconds)
width= Pulse width

Independent Current Source

Syntax: *Name N+ N- isource parameter=value*

N+, N-: nodes

Supported parameters:
type= dc, pulse, sine
freq=Frequency of sinusoidal waveform (Hz)
ampl=Peak amplitude of sinusoidal waveform (Amps)
rise=Rise time for pulse waveform (seconds)
fall=Fall time for pulse waveform (seconds)

period= Period of waveform (seconds)
width= Pulse width

Linear voltage controlled current source

Syntax: *Name N+ N- NC+NC- vccs parameter=value*

N+, N-: source and sink nodes
NC+, NC-: Control nodes

Supported parameters:
gm= transconductance (siemens, A/V)

Linear voltage controlled voltage source

Syntax: *Name N+ N- NC+NC- vcvs parameter=value*

N+, N-: source and sink nodes
NC+, NC-: Control nodes

Supported parameters:
gain=voltage gain (V/V)

Linear current controlled current source

Syntax: *Name N+ N- NC+NC- cccs parameter=value*

N+, N-: source and sink nodes
NC+, NC-: Control nodes,
The control nodes are defined across a current probe.

Supported parameters:
gain=current gain (A/A)

Linear current controlled voltage source

Syntax: *Name N+ N- NC+NC- ccvs parameter=value*

N+, N-: source and sink nodes
NC+, NC-: Control nodes
The control nodes are defined across a current probe

Supported parameters:
gain=trasresistance (V/A, ohms)

HAARSPICE

Nonlinear Components

Diode

Syntax: *Name A C diode parameter=value*
A: Anode
C: Cathode

Supported parameters:
is: Saturation Current (A)
vto: Threshold Voltage (V)
tt: Transit time (sec)
phi: Junction potential (V)
cjo: Zero-bias junction bottom capacitance density (F/m^2)
m: Multiplier
fcp: Forward-bias capacitance coefficient

Bipolar Junction Transistors

Syntax: *Name C B E NPN/PNP parameter=value*
C: Collector
B: Base
E: Emitter

Supported parameters:
area: cross section area (m^2)
is: saturation current (A)
ise: base-emitter leakage current (A)
isc: base-collector leakage current (A)
bf: beta forward

br: beta reverse
nf: forward emission coefficient
nr:reverse emission coefficient
ne: b-e leakage emission coefficient
nc: b-c leakage emission coefficient
vaf: forward Early voltage (V)
var: reverse Early voltage (V)
ikf: forward knee current (A)
ikr: reverse knee current (A)
cje: capacitance, base-emitter junction (F)
vje: voltage, base-emitter junction (V)
mje: b-e grading exponential factor
cjc: capacitance, base-collector junction(F)
vjc: voltage, base-collector junction(V)
mjc: b-c grading exponential factor

cjs: capacitance, collector-substrate junction (F)
vjs: voltage, collector-substrate junction(V)
mjs: c-s grading exponential factor
fc: forward bias capacitance factor
tf: ideal forward transit time(s)
xtf: tf bias coefficient
vtf: tf-vbc dependence voltage(V)
itf: high current factor
tr: reverse diffusion capacitance(s)

Differential Quantizer (quantizer2D)
Syntax: *Name In+ In- Out+ Out- Clk quantizer2 parameter=value*

In+, In-: Quantizer Input
Out+, Out-: Quantizer Output
Clk: Clock input (the clock is not differential)

Supported parameters:
quantizer_vth: Quantizer threshold level (V)
clk_vth: Clock threshold level (V)
vlevel: Output voltage level (V)
trise: Rise time (s)
tfall: Fall time (s)

Miscellaneous Components

Delay
Syntax: *Name In Out delay parameter=value*
In: Delay Input
Out: Delay Output

Supported parameters:
td: time delay (s)

Interface Elements (AtoD, DtoA)
Added Internally
Sign
Added Internally

To get better convergence HAARSPICE adds minimum valued resistors (i.e.GMIN) and Capacitors (i.e cmin).

HAARSPICE SOLVERS

EIGENSPACE LINEAR DOMAIN SOLVER (eld)

The eigenspace linear domain solver (eld) inside HAARSPICE performs time-domain analysis using a Jacobian-free eigenvalue solving algorithm. HAARSPICE-eld gives **~10X** speed up in comparison with uniform time-stepping and adaptive time-stepping algorithms which are used in generic SPICE and other time-domain circuit simulators.

Command

```
eld (filename|-testcircuit) log2(timesteps) run-time
[-savenodes] [-writecircuit]
```

Attributes
filename: Currently "HAARSPICE-eld" supports Cadence Analog Artist generated "spectre like" netlist. The components used are described in the earlier section. All spectre related analyses statements are ignored by HAARSPICE-eld.

variables: It includes two sub-attributes
timesteps order: <4-20> number of timesteps
runtime: time for which the simulation is run

savenodes: By default all the nodes in the netlist are saved. Also includes branch currents or pseudo-nodes which describe the currents in a Modified Nodal Matrix.

writecircuit: Writes out information about circuit

`~/eld/order#,totaltime`
for example a directory generated by running a simulation for 4.098 μs with an order of 15 (2^{15}) is named *"/eld/2E15,4u"*

The data files generated are in ascii text format and have 4 columns of numbers in double float type. An example of a data file

<time>	<value>	<frequency>	<spectral density /w Blackman window>
6.250000000000e-05	4.169160088445e-11	0.000000000000e+00	-5.365224910837e+01
1.875000000000e-04	5.000000005200e-01	2.441406250000e-01	-5.312466206937e+01
3.125000000000e-04	5.000000007146e-01	4.882812500000e-01	-5.500754703673e+01
.....	.....	.....	.....
.....	.....	.....	.....

Example

The 4th Order CT ΣModulator described in the **Component** section is simulated using HAARSPICE-eld for 4.096 μsec with time-step order of 15 (i.e. 2^{15} time steps). The number of time steps or time intervals used in our example equals to the number of clock pulses. The clock is running at 4 GHz with a pulse width of 125ps. Therefore 2^{15} x 125ps = 4.096 μsec

Using the command

```
>>     eld sd4d 15 4096n
```

"sd4d" is the name of the netlist file.

By default the nodes as output files get stored in the directory

```
~/SD4D_new-out/eld/2E15,4u
```

```
Av1:1   Av4:1   I39._inst0:t1  net104   _net2   net56   net72   net86
Av1d:1  E2:1    I39.K0:t1      net112   _net3   net60   net76   net94
Av2:1   E3:1    _net0          net124   net44   net64   net8    net98
Av2d:1  E4:1    net0291        net156   net46   net68   net80   V0:p
Av3:1   E7:1    _net1          net168   net50   net7    net82   V1:p
```

We plot the spectrum of the output node (i.e. columns 3 and 4 from "_net0") of the modulator using gnuplot.

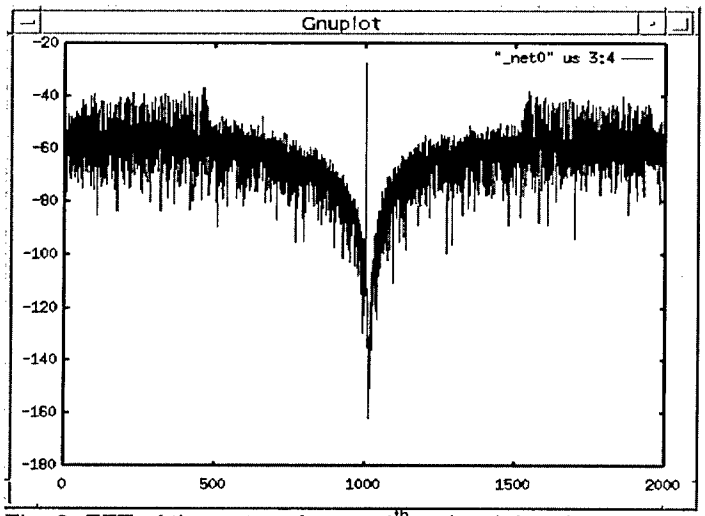


Fig. 3: FFT of the output from a 4th order delta-sigma modulator using a 16384 point Blackman window

HAARSPICE

Performance Metrics

HAARSPICE-eld performance for a 4th order Band Pass Delta-Sigma Modulator circuit was tested on a DELL C800 Laptop running Linux (Red Hat 7.02) operating system.

The Table shown below gives the test results:

HAARSPICE -eld	SPICE/Spectre	Platform
63 seconds	593 seconds	Linux 7.02

In comparison with time-stepping algorithms used in SPICE, HAARSPICE-eld gives **~10X** improvement in speed.

A performance metric giving the measure of accuracy is the **Dynamic Range** of the simulator. To measure the DR of the simulator we perform a two tone test on the 4th order Delta-Sigma Modulator.

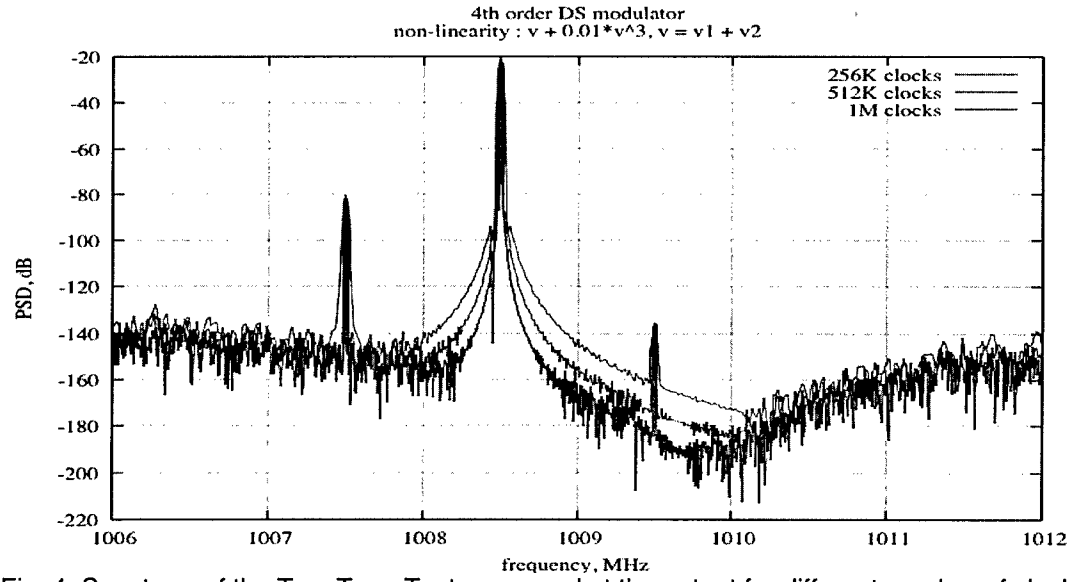


Fig. 4: Spectrum of the Two-Tone Test measured at the output for different number of clock cycles.

The Two Tone Test has one input signal `u1`, at 1008.5 MHz with magnitude of 500mV and a second signal `u2`, at 1007.5 MHz with a magnitude of 0.5mV. The nonlinearity introduced is through the 1st Trans-conductance Cell (i.e. Gm-Cell) of the Modulator. The only other nonlinearity is that of the quantizer. The 1st Gm-Cell nonlinearity generated a 3rd order harmonic is seen at 2x1008.5 – 1007.5 = 1009.5MHz as seen in Fig. 4.

HAARSPICE

Performance Metrics

The Spurious Free Dynamic Range in the Two-Tone test is taken as the Dynamic Range of the simulator.

From Fig. 4, one can compute the DR to be Pmax - Pmin = -20dB - (-200) dB = 180 dB

To resolve the 3rd order Harmonic and get a DR of more than 170 dB it took HAARSPICE-eld 1Million Clock Cycles.

The Table shown below gives the DR with two-tone test:

HAARSPICE (eld routine)	SPICE/Spectre	Clock Cycles
180 dB	173 dB	512 K

This makes HAARSPICE-eld particularly useful in Communication circuits and systems which require high Dynamic Range.

EIGENSPACE EVOLUTION NONLINER DOMAIN SOLVER (ese)

The eigenspace evolution nonlinear domain solver (ese) inside HAARSPICE does time-domain analysis of nonlinear circuits using a Jacobian-free eigenvalue solving algorithm. HAARSPICE-ese gives accurate results when compared with SPICE/Spectre. HAARSPICE-ese code is not yet optimized and therefore we haven't compared it with SPICE or Spectre for speed.

Command

```
ese (-netlist:filename) integration_order
log2(timesteps)run-time [-savenodes] [-writecircuit]
```

Attributes

filename: Currently "HAARSPICE-ese" supports Cadence Analog Artist generated "spectre like" netlist. The components used are the described in the

Component section of this manual. All spectre and verilog-ams related analyses statements are ignored by HAARSPICE-ese.

variables: It includes three sub-attributes
integration order: <2-4> 3 recommended
timestep order: <4-20> number of timesteps
runtime: time for which the simulation is run

savenodes: By default all the nodes in the netlist are saved. Also includes the branch currents or pseudo-nodes which describe the currents in a Modified Nodal Matrix.

writecircuit: Writes out information about the circuit

Example
We choose a 3-latch Quantizer to test ese. Each latch is designed using hbt devices. The Quantizer is clocked with a pulse of 300mV p-p at a period of 250ps (i.e. 4 GHz) and with 50% duty cycle. Below is the Fig. of a single latch circuit.

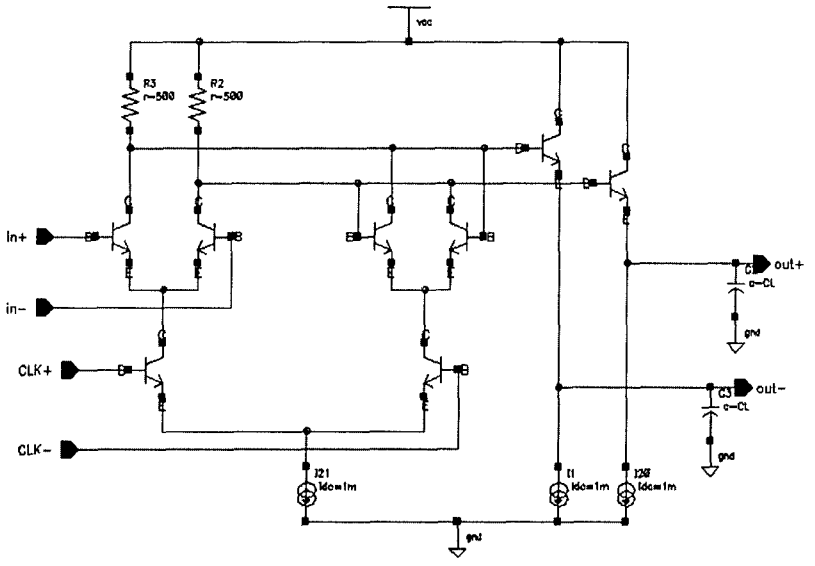


Fig. 3: HBT Latch circuit

The quantizer netlist with the three latches and clock can be found in HAARSPICE examples directory.

Using the command

```
>>     ese NL_latch_test 3 10 10n
```

"NL_latch_test" is the name of the netlist. The integration order used is 3 and the simulation is run for 10ns. It takes approximately 400 seconds to finish the simulation on a DELL C800 laptop running Linux operating system.

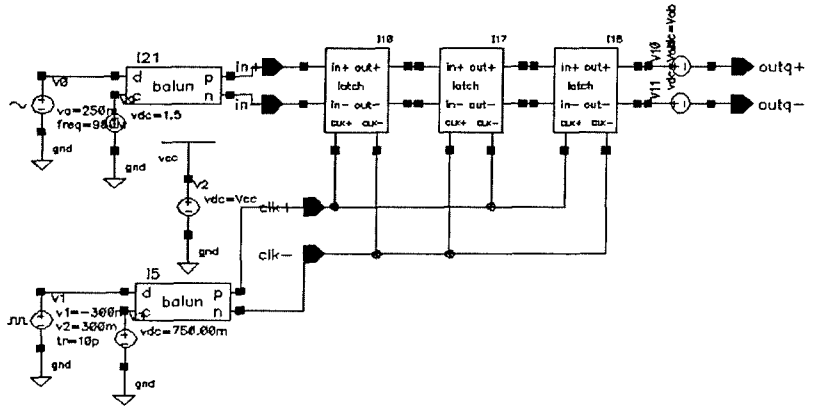


Fig. 4: 3-Latch Quantizer test circuit with input of 250mV at 980MHz

HAARSPICE

The HAARSPICE-ese results are plotted with gnuplot are shown below

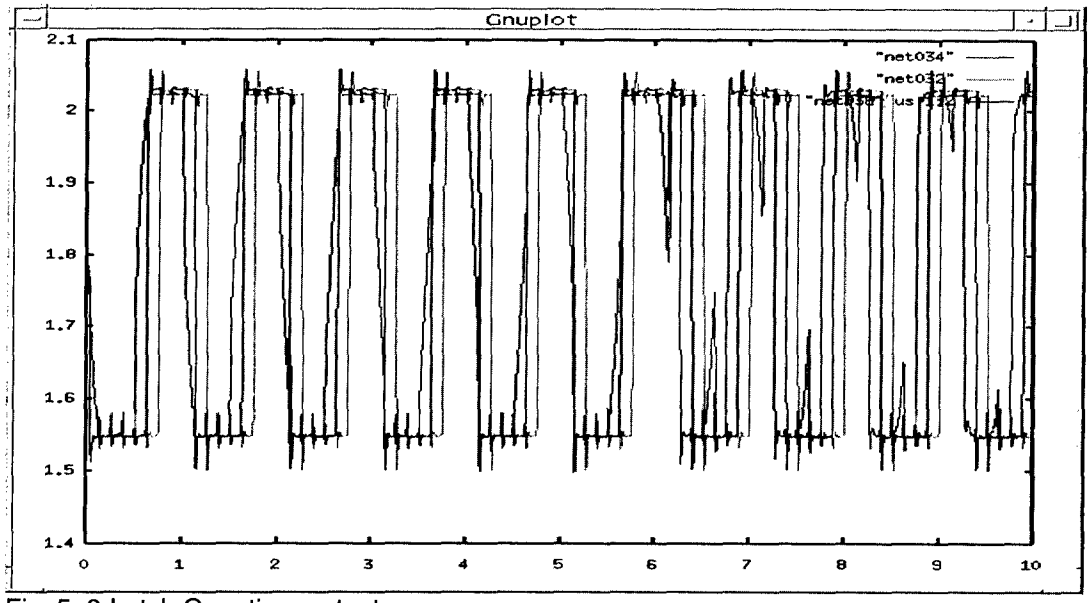


Fig. 5: 3-Latch Quantizer output

Performance Metrics

HAARSPICE-ese performance for a 3-latch Quantizer used in a 4th order Band Pass Delta-Sigma Modulator circuit was tested on a DELL C800 Laptop running Linux (RedHat 7.02) operating system.

The Table shown below gives the test results.

HAARSPICE (ese)	HAARSPICE (enl) Generic Time-stepping code	Platform
400 seconds	> 5000 seconds	Linux 7.02

HAARSPICE-ese is not optimized and does not use a sparse-matrix solver. We are working on adding ARPACK routines to do mathematical computations. We expect the optimized code to give **x10** speed up in comparison with uniform and adaptive time stepping code.

The eigenspace nonlinear solver (enl) inside HAARSPICE does time-domain analysis of nonlinear circuits using a Jacobian-based Newton algorithm. HAARSPICE-enl gives accurate results when compared with SPICE/Spectre. HAARSPICE-enl code is not yet optimized and therefore we haven't compared it with SPICE or Spectre for speed. We are working on adding ARPACK routines to do mathematical computations. We expect the optimized code to have similar speed up in comparison with uniform and adaptive time stepping code.

Command

```
enl (-netlist:filename) integration_order
log2(timesteps)run-time [-savenodes] [-writecircuit]
```

Attributes

filename: Currently "HAARSPICE-enl" supports Cadence Analog Artist generated "spectre like" netlist. The components used are the described in the component section of this manual. All spectre and verilog-ams related analyses statements are ignored by HAARSPICE-ese.

variables: It includes three sub-attributes

integration order: <2-4> 3 recommended
timestep order: <4-20> number of timesteps
runtime: time for which the simulation is run

savenodes: By default all the nodes in the netlist are saved. Also includes the branch currents or pseudo-nodes which describe the currents in a Modified Nodal Matrix.

writecircuit: Writes out information about the circuit

HAARSPICE-enl and HAARSPICE-ese have similar syntax and the same 3-latch Quantizer netlist shown in the earlier section can be used to test HAARSPICE-enl. The simulation with HAARSPICE-enl will be 10X slower in comparison with HAARSPICE-ese.

DC Operating Point (dcp)

Although our goal was to develop a fast time domain simulator, we observed that initial conditions (i.e. DC Operating Point) significantly influenced the time domain simulation results. A fast and accurate algorithm for calculating the DC Operating point was required. We investigated various Homotopy/Continuation methods and found GMIN Stepping and Arc length Homotopy methods to be the most suitable for our purpose. Inside HAARSPICE-dcp we have used both these methods. Except for HAARSPICE-eld, HAARSPICE-dcp is run internally during every time domain simulation to compute the DC operating conditions for nonlinear circuits. By default GMIN stepping is used.

Command

```
dcp (filename)
```

Attributes

filename: Currently "HAARSPICE-dcp" supports Cadence Analog Artist generated "spectre like" netlist. The components used are the described in the Component section of this manual. All spectre and verilog-ams related analyses statements are ignored by HAARSPICE-dcp.

Output: The DC operating point results are saved in files `"filename.bic"` (i.e. Binary format) and `"filename.tic"` (i.e. Text format)

Example

We choose the same 3-latch Quantizer netlist used in the section on HAARSPICE-ese to test HAARSPICE-dcs

Using the command

```
>>   dcp NL_latch_test
```

The HAARSPICE-dcs results are stored inside a binary file "NL_latch_test.bic" and in a text format inside "NL_latch_test.tic"

DC Voltage Sweep (dcs)

DC Voltage sweep inside HAARSPICE was originally written to test Transistor and Diode I-V characteristics and is seldom used to analyze large circuits.

Command

```
dcs (filename)
```

Attributes

filename: Currently "HAARSPICE-dcs" supports Cadence Analog Artist generated "spectre like" netlist. The components used are described in the earlier sections. HAARSPICE-dcs unlike other solvers use the `"dc"` analysis statement from the spectre netlist

Output: The results is stored in a directory with the name `"./filename-out/dcsweep"`

Example

We use a simple Diode circuit to test HAARSPICE-dcs. The diode voltage is swept from -40 to 5 volts. The netlist used can be found in the examples directory.

Using the command

```
>>    dcs diodebvtest
```

The dcsweep results are stored inside `"./diodebvtest-out/dcsweep"`
The directory listing shows the following files.

```
D0:ROut_rs   table   Vdd   VDD:p
```

The `"table"` file list the following contents, which can be used to make plots

Vdd	VDD:p	D0:ROut_rs
-4.000000000000e+01	1.046398822346e+00	-3.895360117773e+01
-3.975000000000e+01	8.032357258910e-01	-3.894676427419e+01
-3.950000000000e+01	5.624481245041e-01	-3.893755187558e+01
-3.925000000000e+01	3.265077005214e-01	-3.892349229956e+01

The output from HAARSPICE-dcs can be plotted using gnuplot

With the gnuplot commands:

```
gnuplot> set data styles lines
gnuplot> plot "diodebvtest-out/dcsweep/table" us 1:(-$2)
```

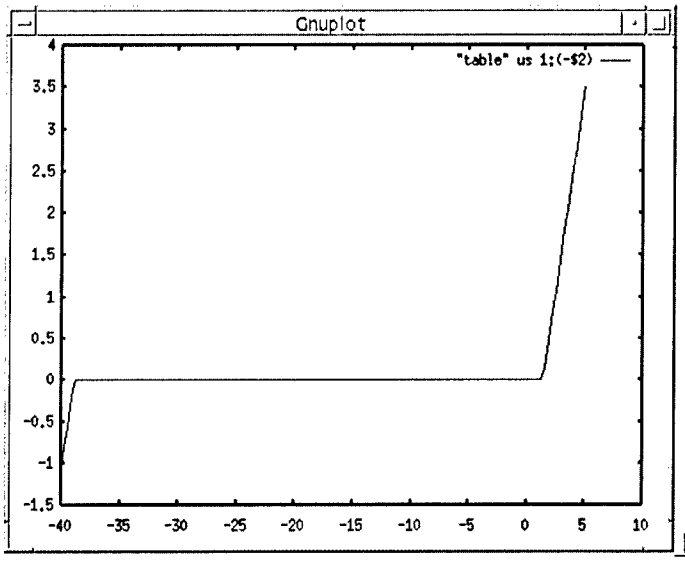


Fig. 6: Plot showing Id versus Vd characteristics

Example Circuits and Error Reporting

The CD supplied with this HAARSPICE User Manual contains the netlists of circuits which can be run and the results plotted with gnuplot.

The user can download gnuplot from the following www site.
`http://www.gnuplot.info/download.html`

The example circuits provided with the CD:

a. 4th Order Continuous Time Band Pass Delta-Sigma Modulator (`sd4d`)
b. 4th Order Continuous Time Band Pass Delta-Sigma Modulator with nonlinearity introduced through input (for two tone tests) (`sd4d_2tone`)
c. 2nd Order Continuous Time Low Pass Delta-Sigma Modulator (`sd2d`)
d. Gummel-Poon Transistor 3-Latch Quantizer (`NL_latch_test`)
e. Simple Diode Circuit for DC Sweep (`diode_test`)

There are ready to use shell scripts available with each of the examples.

Any errors or comments regarding HAARSPICE can be sent to

John Visher (visher@hrl.com)
Rahul Shringarpure (rahuls@hrl.com)
Peter Petre (petre@hrl.com)

Manual of BEMP version 3.0

Sung-Hwan Min* and Madhavan Swaminathan

School of Electrical and Computer Engineering, Georgia Institute of Technology
777 Atlantic Drive, Atlanta, GA 30332-0250, USA

Phone: 1-404-894-3340 Fax: 1-404-894-9959

Email:shmin@ece.gatech.edu, madhavan.swaminathan@ece.gatech.edu

Version 3.0 Sep.08 2003

Broadband efficient macromodeling program (BEMP) is a Windows-based software package that captures the frequency response generated from an electromagnetic tool or measurements and generates both SPICE net list and poles and residues of macromodels for design and simulation of multi-GHz systems as shown in Fig. 1.

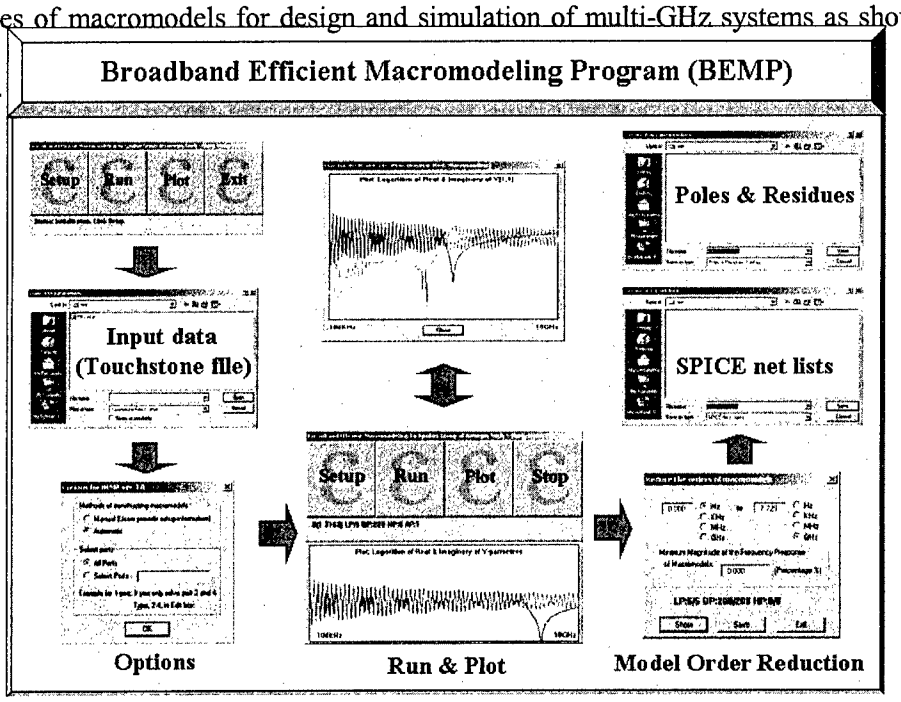
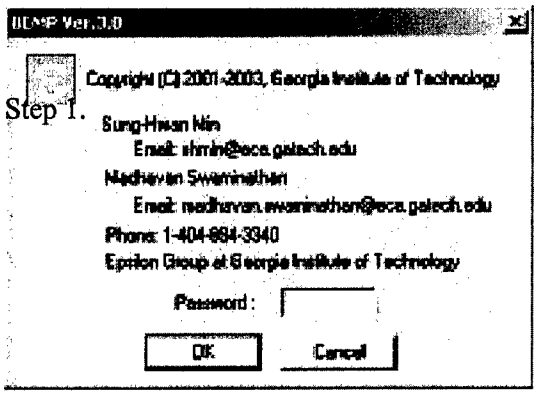
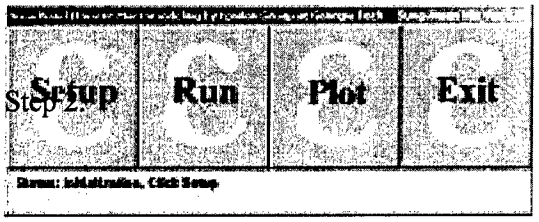


Fig. 1 Broadband Efficient Macromodeling Program



Step 1.

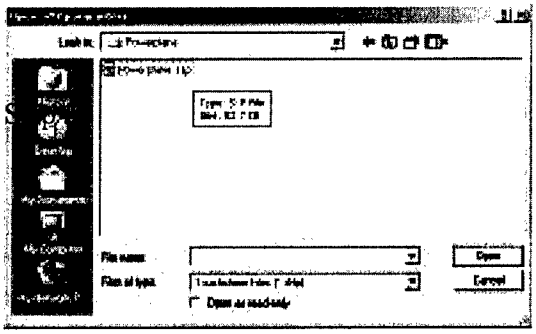
Users provide the password of BEMP. The password can be obtained from Prof. Swaminathan.



Step 2.

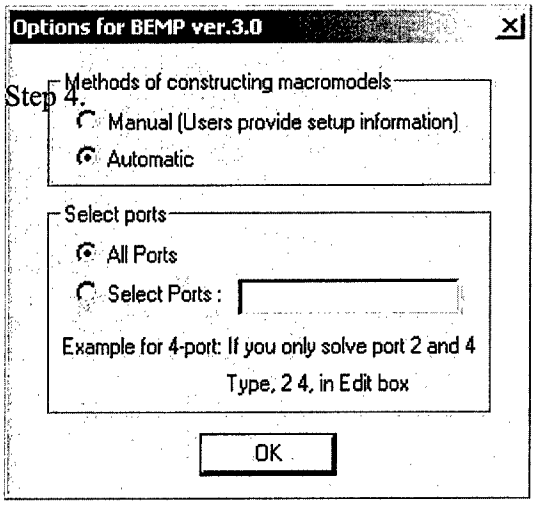
BEMP mainly has four buttons: Setup, Run, Plot, Exit.

Click Setup button.



When users click the Setup button, BEMP prompts to select or type a Touchstone file. The extension of the Touchstone file is sNp, where N represents the number of ports. Please refer to Appendix for details

Click Open button.

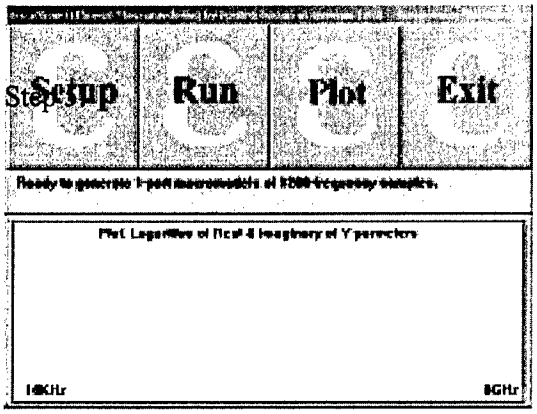


Step 4.

BEMP provides two methods for the construction of macromodels: Manual and Automatic. If users want to use Manual option, please refer to Appendix for details. When users choose Automatic option, BEMP will automatically use its own methods. A developer, Sung-Hwan Min, investigates more robust, sophisticate algorithms for the Automatic option.

BEMP also provide an option for selecting the ports if users want to construct a certain port of macromodels. Please refer to Appendix for details if users want to select the ports.

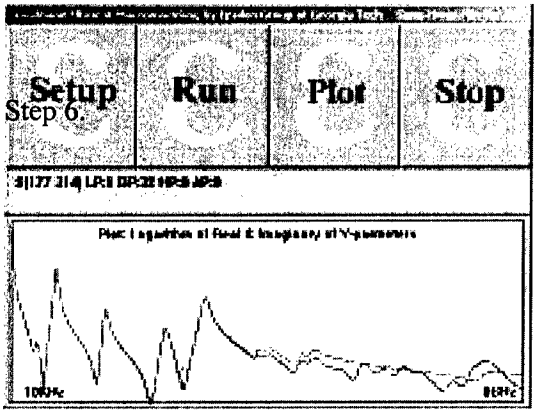
Click OK button.



Step 5.

BEMP loads the original frequency data and show the number of ports and frequency samples. BEMP initializes the window to show the comparison between the original data and the response of macromodels in logarithm scale.

Click Run button.



Step 6.

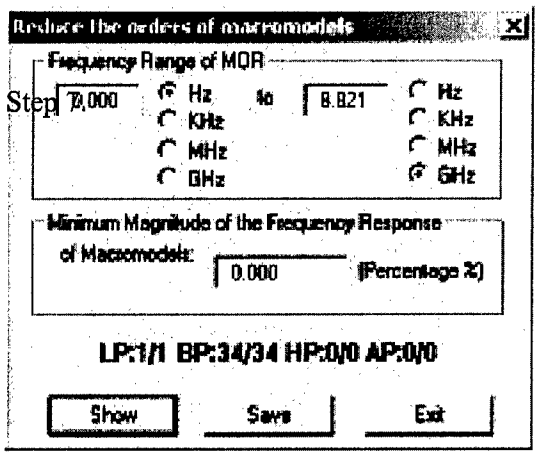
BEMP generates the macromodels by showing the status below Setup button.

S(177 314) says that a subband having samples from 177 and 314 is being constructed.

LP:1 says that BEMP captures 1 low-pass pole. BP:32 says that BEMP captures 32 band-pass poles. HP:0 says that BEMP captures 0 high-pass pole. AP:0 says that BEMP captures 0 all-pass pole.

Note that the order of macromodels is 33.

If users want to stop constructing poles, click Stop button If not, BEMP runs forever.



Step 7.

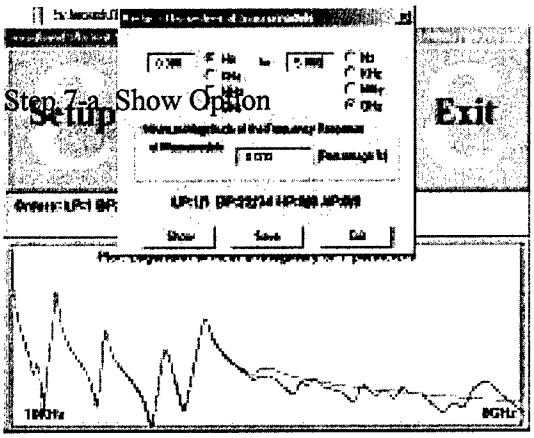
After constructing the macromodels, BEMP gives a function to show and reduce the orders of models. Users can see and reduce the orders by specifying the frequency range of interest or the minimum magnitude of the frequency response.

If Show button is clicked after specifying the options, users can see the performance.

If Save button is clicked after specifying the options, users can save the output of BEMP.

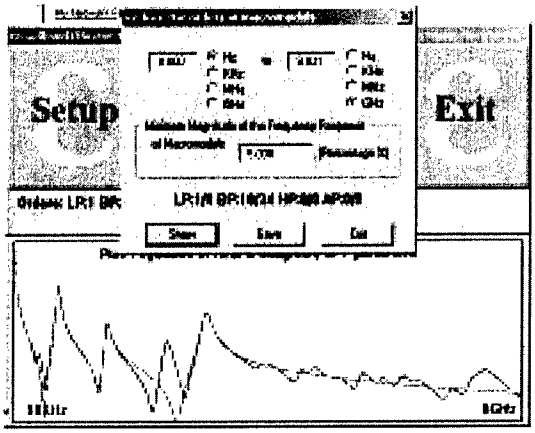
If Exit button is clicked, users can exit.

Step 7-a. Show Option



The spurious or redundant poles are removed by comparing the frequency range with the imaginary value of poles. For example, if users have a range of 5MHz to 2GHz, then all low-pass poles are removed and all band-pass poles located outside the band of interest are removed.

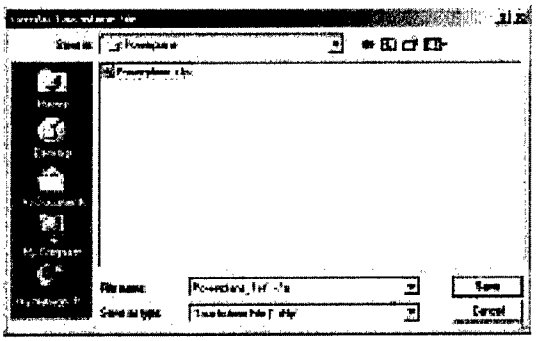
For example provided, users specify the frequency range from 0Hz to 5GHz. The BP order is changed from 34 to 22. Users can see the performance of MOR of BEMP.



BEMP calculates the maximum magnitude of the original data (root-mean-squares). Users can select the minimum magnitude of the frequency response of each macromodel.

For example specified, if users specify the 5% percent of the maximum magnitude of the frequency response, the BP order is changed from 34 to 10. Users can see the performance of MOR of BEMP.

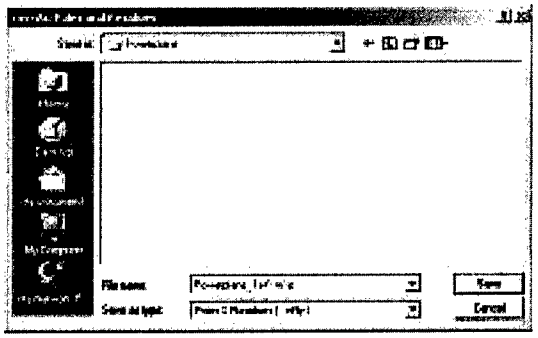
Step 7-b. Save Option



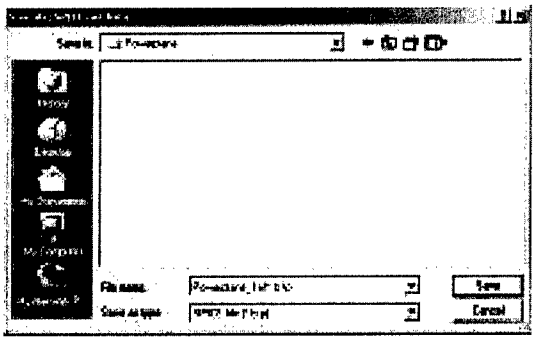
BEMP save the output such as 1) the frequency response of macromodels, 2) the poles and residues of macromodels, and 3) SPICE net lists. In default, the port information is attached in the original filename in order to distinguish the original data and the output of BEMP. For example, "Powerplane.s1p" is changed to

"Powerplane_1of1.s1p". But, users can change the name of the file.

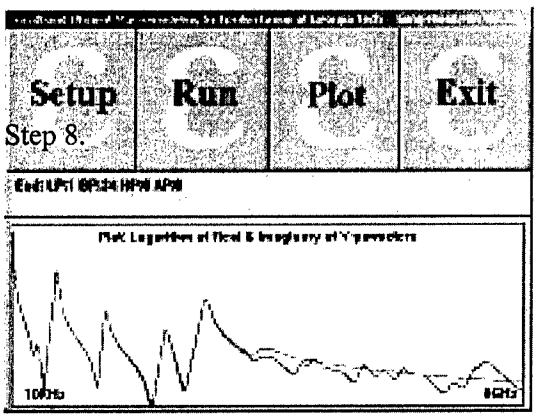
The extension of the file (Touchstone file) having the frequency response of macromodels is the same as the format of a Touchstone file.



The extension of the file (Macromodel file) having the poles and residues of macromodels starts with 'm'.



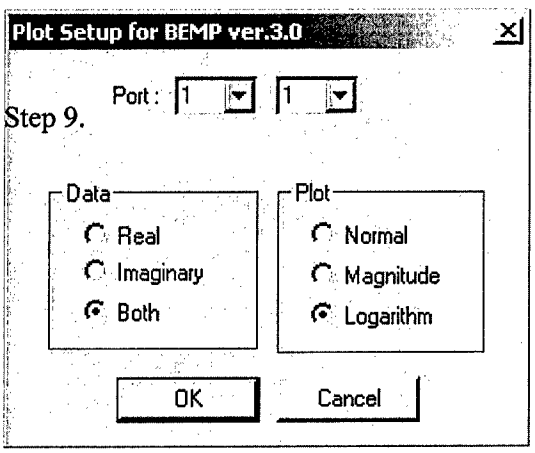
The extension of the file having SPICE net lists starts with 'b'.



Step 8.

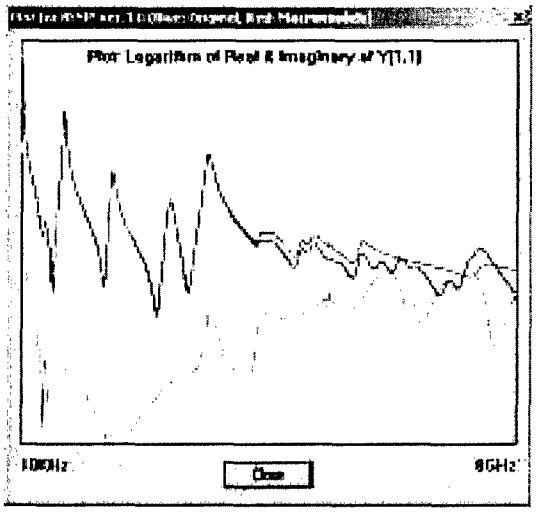
BEMP provide Plot button to show the accuracy of macromodel.

Click Plot button.



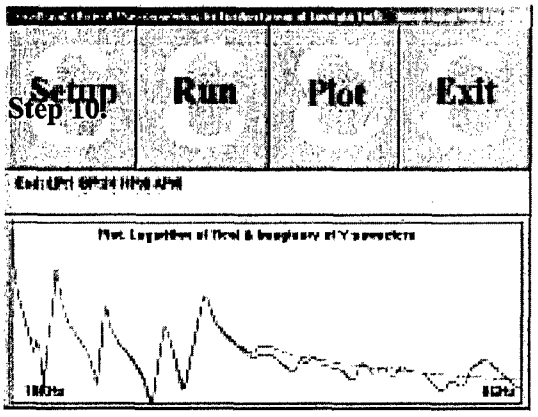
Step 9.

BEMP provides a function to see the output of the frequency response. Users can select the specific port with a specific data and plot formats.



BEMP shows the graphical comparison between the original data and the frequency response of macromodels.

Blue: Original data
Red: Frequency response of macromodels
Green: Error



Step 10:

Click **Exit** button.

Appendix

1. Touchstone file format

 # [Frequency Unit] [Parameter Type] [Data Format] [R Reference Impedance]
Frequency Unit: Hz, KHz, MHz, GHz, THz

Parameter Type: S, Y, Z (S for scattering, Y for admittance, S for impedance)
Data Format: MA, DB, RI (MA for magnitude-angle, DB for dB-angle, RI for real-imaginary)

Reference Impedance: The reference resistance [ohm]
Example 1: **# GHz S MA 50**

Example 2: **# GHz Y RI 1.0**

Examples of Y parameters for four ports in RI data format

HZ Y RI R 1.0

Freq Yr11 Yi11
Yr12 Yi12
Yr13 Yi13
Yr14 Yi14
Yr21 Yi21
Yr22 Yi22
Yr23 Yi23
Yr24 Yi24
Yr31 Yi31
&

2. Manual Option

If users want to use a manual construction, BEMP needs a file containing the construction options. The format used in BEMP is :

Band(A B) Order(C D) Filter(EFGH IJKL) Basis(MNOP) PoleReplacement(Q)

Band(A B): **A** for starting frequency sample and **B** for ending frequency sample of the subband

Order(C D): **C** for the order of numerator and **D** for the order of denominator

Filter(EFGH IJKL): **EFGH** for pre-filter of poles of **broadband** macromodels and **IJKL** for post-filter of poles of **subband** macromodels.

1. **E and I sections**: Users can provide a character **R** for removing the low-pass poles and a dash character for holding the low-pass poles.

2. **F and J sections:** Users can provide several characters: **A, D, K, L, G, H, R, B**

A for not removing the poles
D for removing all poles
K for removing the poles below the starting point of the subband

L for removing the poles below the ending point of the subband

G for removing the poles above the starting point of the subband

H for removing the poles above the ending point of the subband

R for removing the poles within the subband

B for removing the poles below the starting point of the subband or above the ending point of the subband

3. **G and K sections**: Users can provide a character **R** for removing the high-pass poles and a dash character for holding the high-pass poles.

4. **H and L sections**: Users can provide a character **R** for removing the all-pass residues and a dash character for holding the all-pass residues.

Basis(MNOP): **MNOP** for basis functions

1. **M section**: Users can provide a character **L** for the low-pass poles and a dash character for removing the low-pass poles.

2. **N section**: Users can provide a character **B** for the band-pass poles and a dash character for removing the band-pass poles.

3. **O section**: Users can provide a character **H** for the high-pass poles and a dash character for removing the high-pass poles.

4. **P section**: Users can provide a character **L** for the all-pass residues and a dash character for removing the all-pass residues.

PoleReplacement(Q): **Q** for the number of iteration

Example:

Band(1 110)Order(20 21) Filter(RR-- -B--) Basis(LB--) PoleReplacement(1) Band(111 200)Order(20 21) Filter(-R-- -B--) Basis(-B--) PoleReplacement(1) Band(201 300)Order(20 21) Filter(-R-- -B--) Basis(-B--) PoleReplacement(1) Band(301 400)Order(20 21) Filter(-R-- -B--) Basis(-B--) PoleReplacement(1) Band(401 500)Order(20 21) Filter(-R-- -B--) Basis(-B--) PoleReplacement(1) Band(501 600)Order(20 21) Filter(-R-- -B--) Basis(-B--) PoleReplacement(1)

Band(601 700)Order(20 21) Filter(-R-- -B--) Basis(-B--) PoleReplacement(1) Band(701 800)Order(20 21) Filter(-R-- -B--) Basis(-B--) PoleReplacement(1) Band(801 900)Order(20 21) Filter(-R-- -B--) Basis(-B--) PoleReplacement(1) Band(901 1000) Order(20 21) Filter(-R-A -G--) Basis(-B-A) PoleReplacement(1)

Example:

Band(1 100)Order(10 11) Filter(RRRR ----) Basis(LB--) PoleReplacement(2) Band(101 200)Order(10 11) Filter(-K-- -G--) Basis(-B--) PoleReplacement(2) Band(201 300)Order(10 11) Filter(-K-- -G--) Basis(-B--) PoleReplacement(2) Band(301 400)Order(20 11) Filter(-K-- -G--) Basis(-B--) PoleReplacement(2)

Reference:

BEMP uses efficient constructions methods: 1) multiport passivity formulae [2,3], 2) band division [2,3], 3) subband reordering [2,3], 4) subband dilation [2,3], 5) pole replacement [2,3], 6) automatic construction sequences, and so on. BEMP uses the least squares approximation (version 1.0, version 2.0, version 3.0) [4] and vector fitting [5] (version 3.0) for approximating the frequency response.

[1] **S. H. Min** and M. Swaminathan, "Efficient constriction of two-port passive macromodels for resonant networks," IEEE 10th Topical Meeting on EPEP, Oct. 2001.

[2] **S. H. Min** and M. Swaminathan, "Construction of broadband passive macromodels from frequency data for simulation of distributed interconnect networks," IEEE EMC-Zurich, Feb. 2003.

[3] **S. H. Min** and M. Swaminathan, "Construction of broadband passive macromodels from frequency data for simulation of distributed interconnect networks," accepted for publication on IEEE Trans. Electromagnetic Compatibility, 2003.

[4] K. L. Choi and M. Swaminathan, "Development of model libraries for embedded passives using network synthesis," IEEE Trans. ADSP, vol. 21, no. 1, Feb. 1998. [5] B. Gustavsen and A. Semlyen, "Rational approximation of frequency domain responses by vector fitting," IEEE Trans. Power Delivery, vol.14, no. 3, July 1999.

7.3 Published Papers and Presentations

1. R. Shringarpure, T. Kaplan, G. Valley, P. Petre, and J. Visher,"Modeling Thermal Induced Inter-Symbol Interference of Feedback DACs in Delta-Sigma Modulators",BMAS 2004, 21-22 October 2004.
2. S. D. Gedney, A. Zhu, W.-H. Tang, G. Liu, and P. Petre," A Fast, High-Order Quadrature Sampled Pre-Corrected FFT for Electromagnetic Scattering," Microwave and Optical Technology Letters, vol. 36, no. 5, pp. 343-349, March 5 2003.
3. A. Zhu and S. D. Gedney, "A Quadrature Sampled Pre-Corrected FFT Method for the Electromagnetic Scattering from Inhomogeneous Objects," *IEEE Antennas and Wireless Propagation Letters*, vol. 2, no. 1, pp. 50-53, 2003.
4. S. Min and M. Swaminathan, "Efficient Construction of Passive Macro-models for Resonant Networks", Proceedings of the 10th Topical Meeting on Electrical Performance of Electronic Packaging, Boston, Massachusetts, Oct. 2001.
5. S. Min and M. Swaminathan, "Construction of broadband passive macro-models from frequency data for simulation of distributed interconnect networks", Proceedings of the EMC Zurich '03, ETH Zentrum, Zurich, Switzerland, Feb. 2003.
6. S. Min and M. Swaminathan, "Construction of broadband passive macro-models from frequency data for distributed interconnect networks", published in the *IEEE Transactions on Electromagnetic Compatibility*, Nov 2004.
7. R. Mandrekar and M. Swaminathan, "Delay Extraction from Frequency Domain Data for Causal Macro-modeling of Passive Networks", Submitted to the International Symposium on Circuits and Systems 2005.
8. R. Mandrekar, M. Swaminathan, Sungjun Chun, "Application of wavelets and generalized pencil-of-function method for the extraction of noise current spectrum and simulation of simultaneous switching noise," Proceedings. 17th International Conference on VLSI Design, 2004 Pages:995 – 1000
9. B. Mutnury, M. Swaminathan and J. Libous, "Macro-Modeling of Non-Linear I/O Drivers using Spline Functions and Finite Time Difference Approximation", Proceedings of 12th Topical meeting on EPEP, Oct 2003. Pages:273 – 276
10. B. Mutnury and M. Swaminathan, "Macro-Modeling Methodology For Weakly Non-Linear Digital Drivers", Proceedings of SPI workshop 2003, Italy.
11. Bhyrav Mutnury, Madhavan Swaminathan and James Libous, "Black-Box Modeling of Non-linear I/O Drivers," The Applied Computational Electromagnetic Society (ACES), Syracuse, NY, April 19-23, 2004
12. Bhyrav Mutnury, Madhavan Swaminathan and Jim Libous, "Modeling of Power Supply Noise using Efficient Macro-Model of Non-Linear Driver," accepted for 2004 IEEE International Symposium on EMC, San Jose (CA), USA, August 9-13, 2004, Pages: 988-993.
13. Bhyrav Mutnury, Madhavan Swaminathan and Jim Libous, "Macro-Modeling of Non-Linear Digital I/O Drivers," accepted for *IEEE Transactions on Advanced Packaging*, Feb' 2005.
14. Bhyrav Mutnury, Madhavan Swaminathan, Moises Cases, Nam Pham, Daniel N. de Araujo and Erdem Matoglu, "Macro-Modeling of Transistor Level Receiver Circuits," accepted at 13th Tropical Meeting on Electrical Performance of Electronic Packaging (EPEP 2004), Portland, Oregon, October, 2004 **(Received Intel Best Student Paper Award)**.
15. R. Mandrekar, M. Swaminathan, "Wavelet based nonparametric regression approach for de-noising and modeling of transient switching noise measurements", Electronics Packaging Technology, 2003 5th Conference (EPTC 2003), Dec. 2003 Pages:39 – 44.